References to Graphical Objects in Interactive Multimodal Queries

Daqing He
何大庆

Doctor of Philosophy
Institute for Communicating and Collaborative Systems
Division of Informatics
University of Edinburgh
2000

Abstract

This thesis describes a computational model for interpreting natural language expressions in an interactive multimodal query system integrating both natural language text and graphic displays. The primary concern of the model is to interpret expressions that might involve graphical attributes, and expressions whose referents could be objects on the screen.

Graphical objects on the screen are used to visualise entities in the application domain and their attributes (in short, domain entities and domain attributes). This is why graphical objects are treated as descriptions of those domain entities/attributes in the literature. However, graphical objects and their attributes are visible during the interaction, and are thus known by the participants of the interaction. Therefore, they themselves should be part of the mutual knowledge of the interaction.

This poses some interesting problems in language processing. As part of the mutual knowledge, graphical attributes could be used in expressions, and graphical objects could be referred to by expressions. In consequence, there could be ambiguities about whether an attribute in an expression belongs to a graphical object or to a domain entity. There could also be ambiguities about whether the referent of an expression is a graphical object or a domain entity.

The main contributions of this thesis consist of analysing the above ambiguities, designing, implementing and testing a computational model and a demonstration system for resolving these ambiguities. Firstly, a structure and corresponding terminology are set up, so these ambiguities can be clarified as ambiguities derived from referring to different databases, the screen or the application domain (*source ambiguities*). Secondly, a meaning representation language is designed which explicitly represents the information about which database an attribute/entity comes from. Several linguistic regularities inside and among referring expressions are described so that they can be used as heuristics in the ambiguity resolution. Thirdly, a computational model based on constraint satisfaction is constructed to resolve simultaneously some reference ambiguities and source ambiguities. Then, a demonstration system integrating natural language text and graphics is implemented, whose core is the computational model.

This thesis ends with an evaluation of the computational model. It provides some concrete evidence about the advantages and disadvantages of the above approach.

Acknowledgements

My special thanks to my two supervisors, Dr Graeme Ritchie and Dr John Lee, for taking me into the area of natural language processing and intelligent multimodal interfaces and devoting enormous amount of time and effort to the research that culminated with this thesis. Their comments were always detailed and precise, which were sometimes painful to accept. Their tremendous patience to my English is also greatly appreciated.

My thanks then to Dr Frank Keller for giving advice on experiment design and statistics; Angus Maclean at SCMS, the Robert Gordon University, and my wife Hua Cheng for doing the tedious grammar checking. Thanks too to other peers who helped me in my PhD study: Jessica Chen-Burger, Jeremy Crowe, Chris Gathercole, Ion Androutsopoulos, other folks in E17 and F11 and the 14 volunteers who helped me in the evaluation.

I acknowledge the economic support of a Colin & Ethel Gordon Scholarship from the Faculty of Science and Engineering, the University of Edinburgh and a British ORS award during my three years of PhD study.

I would also like to thank members of the IR group at SCMS, the Robert Gordon University, especially Dr Ayşe Göker, for their support, encouragement and tolerance while I was writing up the thesis and working on a project at the same time.

During these years I had the moral support of my family both in China and here in Edinburgh. I owe great debt to them, for their tremendous love, support and patience, especially those from my wife Hua. I'd like to dedicate this thesis to them.

Declaration

I hereby declare that I composed this thesis entirely myself and that it describes my own research.

Daqing He
Edinburgh
August 2, 2000

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 What is this thesis about?

With the development of computer technology, more and more computer systems used in human computer interactions integrate more than one communication modality (e.g. graphics, natural language and gesture), and these are usually called **multimodal systems**. When the systems have the aim to improve the efficiency, effectiveness and naturalness of human-machine interaction by representing, reasoning, and acting on models of the user, domain, task, discourse and modality, they are **intelligent systems** [Maybury and Wahlster, 1998, p. 1].

This thesis describes the design of a computational model of analysing referring expressions in an intelligent multimodal system. In the context of this thesis, an intelligent multimodal system more specifically refers to the system which integrates natural language inputs with corresponding graphic display on the screen, and the graphical objects on the screen are used to visualise entities in the application domain and their attributes i.e. domain entities and domain attributes. The principal concern of the model is that it should be capable of dealing with referring expressions involving visual display attributes, and referring expressions whose referents are not merely the entities in the application domain, but the items on the screen. Following terminology in previous research, by **a referring expression** we mean a linguistic form that is used to select a particular entity in some real or imaginary world [Dale, 1992, p. 1]. We define referring expressions involving visual display attributes and referring expression whose referents are screen entities as **visual references**. The intended meanings of

other terms, like visual display attributes and screen entities, will become clear later in this thesis. The reasons for handling those types of referring expressions will be presented in Section §1.2.

In order to demonstrate the utility of the model in analysing those types of referring expressions, we will describe an intelligent multimodal system that provides the appropriate infrastructure around the model. This system, named *Intelligent Multimodal Interface employing Graphics (IMIG)*, integrates both natural language and graphic modalities in its input and output. The content and structure of IMIG are intended to provide the necessary information for the reference resolution process.

1.2 Motivations

1.2.1 The problems

World model and display model

The goal of Human-Computer Interaction (HCI) is to make interactions between humans and computers as natural as that between humans. According to Bunt's *Multimax Principle*, the characteristics of human communication is that "*the participants use all the modalities and media that are available in the communicative situation.*"[Bunt, 1998] Therefore, Human-Computer Interaction will involve more and more multimodal and multimedia information.

An intelligent multimodal system is a knowledge based system, where the knowledge bases are designed to provide different aspects of knowledge. This arrangement maintains the modality and the portability of the system. In this section, we are interested in two knowledge bases. The first one is the **world model**, which specialises in providing facts about the represented world (i.e., the application *domain* of the system). It usually includes knowledge about which entities are in the domain, what their attributes are and what their relations to other entities are. In this thesis, we call the entities in the world model the **domain entities**, and their attributes the **domain attributes**.

It is common for a system presenting information in visual form on the screen (usually in graphic form, e.g. a GIS [Antenucci et al., 1991]) to have a **display model**. This

model is the second database that we are interested in, and it has explicit representation of which items are currently on the screen and what their attributes are. Its role varies among systems, but is mainly to maintain the visual display in an orderly fashion and to connect visual objects to domain objects. It must be updated systematically as objects appear, disappear or are moved on the screen. Very often, the display model performs low-level housekeeping work for the display.

An imaginary system

Many example dialogues in this thesis are drawn from an imaginary multimodal knowledge base system employing both natural language and graphics. The system, which is called the **Car Selection System**, can be seen as an online interactive catalogue displaying details of cars in a stock for sale. These are specific cars, not models of cars. In this respect, the system resembles a used-car salesroom. As shown in Figure 1.1, icons (symbols) in the DISPLAY area represent individual cars, for example, there might be a blue icon representing a particular Nissan 1993 saloon. Various characteristics of the icons convey attributes of the corresponding cars, for example, the colour of the icon might indicate the year of manufacturing of that particular car. As such, the colours of the icons may not be the same as the real colour of the cars. A table of how various attributes of the real cars are represented is displayed in the KEY area. The dialogues involve a customer who is browsing through cars with a view to buying. The POTENTIAL BUY area is sometimes used by the user to collect the icons of cars that he/she is interested in. During the interaction, the user can ask about the cars on the screen, or perform actions, such as move, remove and add on the icons of those cars. (1.1) demonstrates an example interaction between a customer and the system.

(1.1) **User:** What is the insurance group of the green car? (a)
System: `It is group 5.` (b)
User: Move it to the potential buy area. (c)
System: `The green car has been moved.` (d)

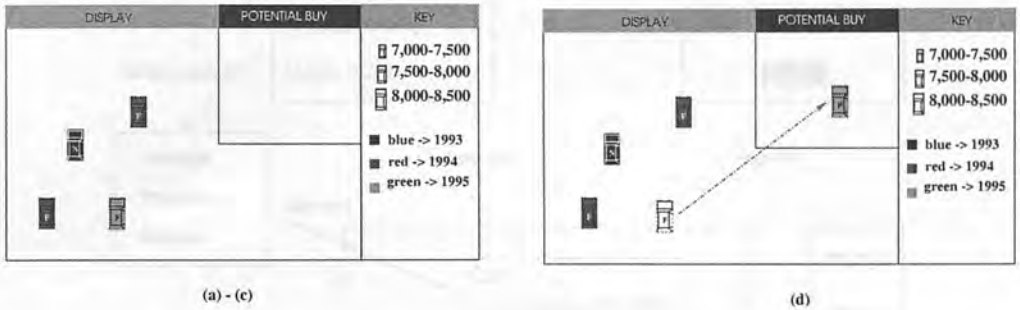


Figure 1.1: Screen displays for (1.1) of the Car Selection System

Some reference processing problems

Graphic entities on the screen, simple or complex, are geometric entities. They have their own attributes, like colour, shape, size, position on the screen, etc. These attributes are visible to and are thus known to both participants of the interaction as long as the entities are on the screen. We call them **visual display attributes**. As part of the mutual knowledge of the participants, the visual display attributes may be included in referring expressions [Hawkins, 1978], and a system employing natural language and graphics should be able to interpret these expressions.

Most previous systems employing natural language and graphics either do not consider visual references or assume that "*graphics on the screen are merely descriptions*" in their generation systems [André and Rist, 1994] (detailed discussion of these systems can be found in Chapter 2). In the second case, visual display attributes could be generated in a referring expression. This is illustrated in Figure 1.2. A domain entity $\mathbf{car}_1$, which is depicted by a blue block on the screen, can be referred to by the phrase "`the blue car`", where "`blue`" is a visual display attribute. However, a visual reference depending entirely on the visual display attributes is beyond those systems' abilities. For example, the system cannot link the phrase "`the blue block`" with the same entity $\mathbf{car}_1$, if both the "`blue`" and the "`block`" are visual display attributes,

A system that has to understand rather than to generate such phrases would face even more difficult problems.

The first problem that the reference processing mechanism has to face is to determine

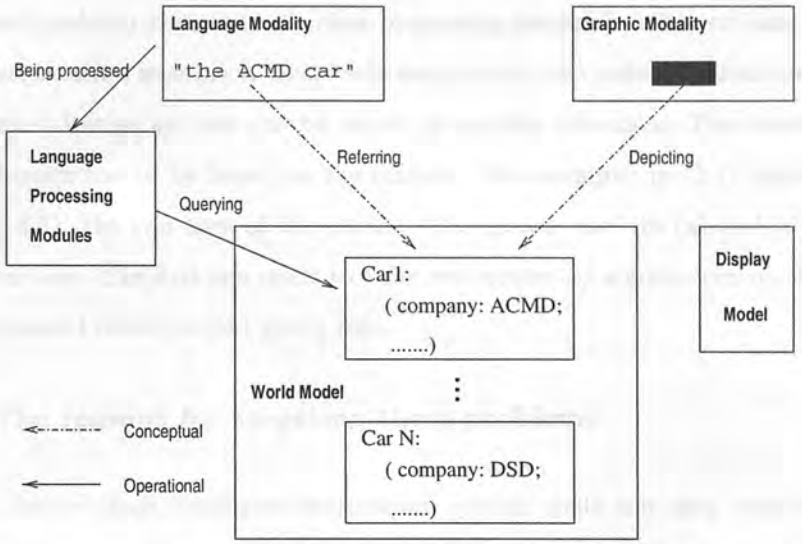


Figure 1.2: The knowledge representation about an entity under the assumption that graphics on the screen are merely descriptions. The display model is not among the knowledge used for language processing.

whether or not an attribute used in a referring expression is a visual display attribute. At a first glance, this may sound trivial. However, it is not the case if considering the different attributes bearing the same names in the domain and on the screen (i.e., colours, sizes, and positions can exist both in the domain and on the screen), and the representation relations between these two sets of attributes. For example, a car that is 3.5 meters long (i.e., a size attribute in the application domain) may have an icon that is 2 cm long(i.e., a size attribute on the screen); or a car worth 2,200 Pound Sterling may have a green colour attribute on the screen, while its real colour in the domain is dark red. In this case, this car can be called not only by a common reference "`the dark red car`", but also by a visual reference "`the green car`", depending on the context. There is not a simple rule saying that a dark red car can also be called "`a green car`", because other dark red cars may have an icon in another colour.

The problem can be shown from another angle. If the domain also contains green cars, the interpretation of the phrase "`the green car`" has to be able to answer the question: "*is it a dark red car with a green icon, or a green car with an icon in whatever colour?*"

The second problem that the reference processing mechanism has to face is that a referring expression, whether a visual reference or not, can refer to a domain entity in one situation, but to an icon on the screen in another situation. The determination of the reference has to be based on the context. For example, in (1.1) (accompanied by Figure 1.1), the two uses of the phrase "`the green car`" in (a) and (d) refer to different entities. The first one refers to a *car* represented by a green icon on the screen, while the second refers to that green *icon*.

1.2.2 The reasons for targeting these problems

Would it matter if an intelligent multimodal system could not deal with the above problems? These problems could disappear if there is no graphic display for domain entities, or there is no natural language input, or there is a restriction that the user cannot use attributes of graphic entities for referring. However, as most intelligent multimodal systems aim at providing users with natural interactive environment resembling communications between human beings, such restrictions are inappropriate.

These problems cannot be ignored. Without a clear and correct referent for each referring expression in a query, it is impossible to infer the correct meaning of the query and produce the right result. That is, ignoring these problems could result in an unnatural communication environment.

These problems cannot be resolved in simple ways either. They are essentially due to the existence of two sets of related attributes and entities in multimodal systems. As shown in the examples, without considering the environment, i.e., what items are on the screen and what objects are in the domain, along with other aspects, such as the context of the dialogue and the user's query manner, etc., it is difficult to find a satisfactory solution.

These problems (or at least some of them) cannot be avoided by using smart interface design. The graphical objects on the screen are the results of viewing some domain entities from a particular angle for a specific purpose. Therefore, different graphical objects may result from projections for different purposes. In addition, it is often the case that, after a projection, some aspects of the domain entities are simplified,

whereas others are elaborated by visualization, as reflected by the graphical objects. This demonstrates that there are needs for graphical objects to appear differently from what the corresponding domain entities look like, which is irrelevant to how smart the interface is.

The discussion in Section §1.2.1 demonstrates that it is this complex relation between a domain entity and its corresponding graphical object that causes the problems mentioned above. An appropriate method for handling these problems is to provide the system with the knowledge about the graphical objects, the knowledge about the relations between the domain entities and the screen entities, and the ability to use such knowledge in the interpretation process.

1.2.3 More evidence from the literature

The above problems are not unfamiliar in the literature. Some aspects of the problems have been mentioned in various places, though no solution has been provided. [Ben Amara et al., 1991] discusses natural language expressions using the graphical features of objects to refer to those objects. Their scenarios involve querying and displaying a local area network. One of their examples related to our problems is that the user can type in one of the following sentences when he/she wants to remove a workstation shown as a purple icon:

1. remove the stellar workstation.
2. remove the purple workstation.
3. remove the purple icon.

Ben Amara et al. think that the second command is ambiguous because it is not clear whether it is the workstation in the application domain or the icon representing the workstation that is purple.

[Binot et al., 1992] enumerates some examples of natural language references using graphical spatial relations in an application domain that is the same as that of [Ben Amara et al., 1991]. Two such examples are "*add a PC to its left*" and "*remove*

the left workstation" They think that the spatial relations in both cases are ambiguous as whether the relations are in the domain or in the graphical representation.

Both papers briefly suggest that the ambiguities must be resolved and can be resolved by using the context of the utterance, or by asking the user to disambiguate explicitly. In addition, the resolution process must have access to a representation of the spatial geometry of the domain.

1.3 Our goal, approach and achievements

1.3.1 Our goal

The problems mentioned in Section §1.2.1 can be summarised as problems relevant to one situation, that is, "*graphics on the screen, their attributes, and their relations with the domain entities could be referred to or used during the interaction*". The reasons that they are interesting research problems are:

1. all the screen related information, i.e., the identities of screen items, their attributes and their relations with other entities are not explicitly represented, or are represented but cannot be accessed by the language processing modules in previous systems;

2. although some researchers have noticed these problems (see the discussion in Chapter 2), none of them provides a method that can effectively and systematically resolve these problems.

The **goal** of this thesis is to provide a computational model for interpreting referring expressions involving both domain and graphics properties so as to deal with the above problems effectively and systematically.

1.3.2 Our approach

General remarks and simplifications

Our approach has two features as determined by the stated goal. First, we do not attempt to design a pure linguistic theory. What we aim at is a computationally

realisable method or model, which is theoretically motivated and draw heavily on theories in pure linguistics [Halliday, 1994].

Secondly, the outcome of our approach, the computational model, obviously has to have the ability of resolving other common referring problems. We attempt to use a constraint satisfaction method, a method that has been used previously in resolving reference [Mellish, 1985, Haddock, 1988], for both reference disambiguation and the resolution of the problems mentioned in Section §1.2. However, being our primary interest, the latter will be the focus of this thesis.

As the first attempt to resolve the problems mentioned in Section §1.2, we only concentrate on related aspects of particular relevance. Therefore, several simplifications were made in our current approach:

- In the linguistic aspect, the types of input sentences are restricted to yes-no questions, wh-questions and imperative commands. Declarative sentences are excluded because they could cause complex database operations like truth maintenance, which is far from the core of our research. The noun phrases in the input sentences are restricted to those referring to a singular concrete entity which exists in the system's knowledge bases. The pre-modifiers of the noun phrases are epithet adjectives and/or a noun classifier [Halliday, 1994], and the only type of post-modifier allowed is prepositional phrases. In addition, we assume that there is no *one* anaphora and ellipses in the input sentences.

- In the process of resolving ambiguities, we assume that the model to be built does not have to resolve a referring expression that does not have a referent in the system, which could be arguably too strong (more detail about this part of simplification can be found in Chapter 5).

- In the knowledge representation, we assume that there is a one-to-one mapping between an icon and its corresponding domain entity. Each icon on the screen represents one and only one domain entity, and a domain entity, if shown on the screen, has only one icon.

- In the interface aspect, especially in the implemented IMIG system, the user

input is assumed to be typed-in text with or without pointing actions[1]. We have not considered complex use of pointing devices. In the car selection domain, end users' operations are assumed to work only on the screen. That is, they cannot change the content of the world model.

- Although the system can model the user's state by noticing whether or not an object is mentioned in a previous dialogue, any more complicated functionality, such as modelling the user's intention, is beyond the ability of the IMIG system.

As far as we know, there is no available corpus relevant to our topic. All the heuristics proposed in this research are based on hand analysis of several dialogues imagined by the author. From this point of view, evaluation presented in Chapter 7 is essential for providing a verification of our modal.

Source and source ambiguities

As noted, one reason that an intelligent multimodal system might lack the ability of interpreting visual references is that the visual display attributes and relations, which are essential for the interpretation, are either not explicitly represented in the system or not accessible by the language processing modules. Therefore, the first step in our approach is to revise the knowledge representation scheme of the system to explicitly represent visual display attributes and relations and make them accessible to the language processing modules. This step reflects our belief that the graphics on the screen are stand-alone geometric entities rather than merely descriptions, and they are parts of the mutual knowledge that is shared by the two participants of the dialogue. Consequently, the system, as one participant in the interaction, has to represent and be able to access them.

In the new knowledge representation scheme shown in Figure 1.3, the graphic entities on the screen and their attributes are represented in the *display model*, just as the domain entities and their attributes are represented in the *world model*. Now, the two sets of entities and attributes can be distinguished by the places where they are represented.

[1] In the future, it is possible that practical IMIG systems will use speech input, but at present, it is reasonable to explore the basic semantic issues using keyboard input.

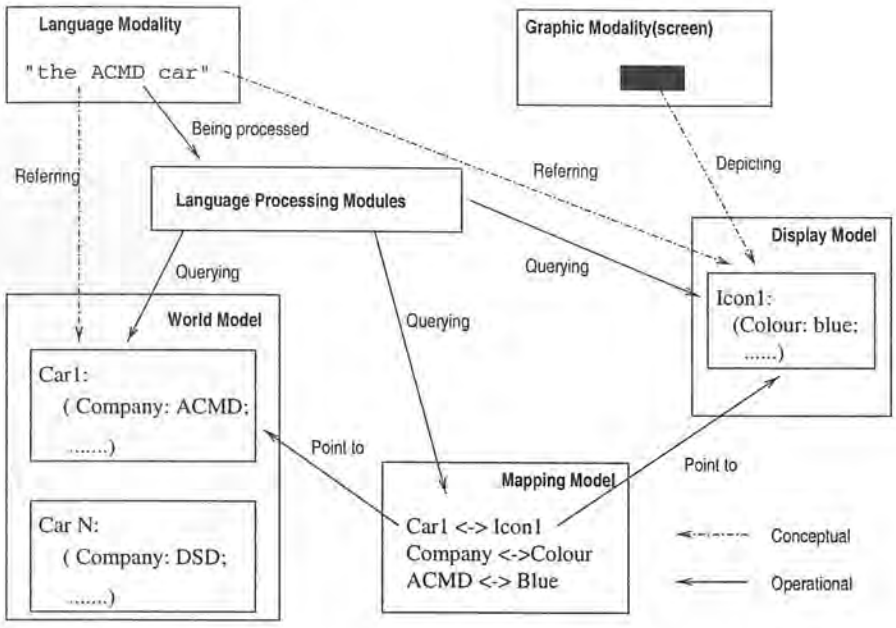


Figure 1.3: The new knowledge representation scheme to facilitate the system's ability to deal with visual references. Both the display model and the mapping model are used as resources for language processing.

If we adopt the term **source** for the particular model where an attribute, entity or action is represented, the graphic entities in the display model can then be referred to as having the *screen* source, and thus are called the **screen entities**. Similarly, the entities in the world model have the *domain* source, and are referred to as the **domain entities**. Accordingly, the attributes of the entities and the actions applicable to the entities are divided into the **screen/domain attributes** and the **screen/domain actions**.

In principle, more than one graphical display can be used to visualise different projections of the application domain at the same time. Each graphical display, in this case, can act as the source of the entities/attributes/relations represented in that display. Although we assume that there is only one graphical display and only two sources in an intelligent multimodal system, we do not expect to have difficulty in extending the essential ideas in this thesis to the situations with more than two sources.

Using the source concept, we can see that the problems mentioned in Section §1.2.1 all relate to one question, that is whether the sources of the attributes, entities, and

relations that arise in the referring expressions or sentences, are the screen or the domain. So, we call them the problems of **source ambiguities** (more detailed discussion is in Chapter 3).

Although it is important to distinguish between a screen entity and a domain entity, for example, the former may have a "`colour`" attribute even if the domain makes no use of the attribute, various processes such as reasoning could be simplified if we represent both entities using some uniform schemata, but allowing for an explicit relation (of denotation) between the screen symbol and the corresponding domain entity. This relation shall be referred to as the **representation mapping**. The mapping can be between the attributes of graphic entities such as shape, colour, size etc. and the same type of attributes of the domain entities, e.g. the shape represents the shape, and the colour represents the colour. Sometimes, the visual attributes can also be used to stand for domain attributes which are not directly visual, e.g. the shape represents the price, the size represents the population, etc.. These mappings are stored in a space called the **mapping model**. This is accessible to the language processing modules, because these representation relations are needed both in the resolution process and probably during the interaction (see Figure 1.3). The screen symbol and the domain entity connected by this mapping are called **corresponding entities** of each other.

MRL_S

Before a source ambiguity is resolved, the source of an entity/attribute/relation is ambiguous, while after the resolution, the source of this entity/attribute/relation is identified to be either the screen or the domain. This requires the meaning representation language used in the system for conveying the meaning of the input sentences to be able to cope with both situations. For this purpose, we developed a logic-based meaning representation language called **MRL_S**. It is composed of *QLE* and *LE*, two layers dedicated for the meaning representation before and after disambiguation. The development of the MRL_S is the second step of our approach. More about this aspect will be presented in Chapter 4.

Constraint satisfaction resolution method

The third step is to use a constraint satisfaction resolution method in order to resolve the source ambiguities and other referential ambiguities together. We treat various resources for the source assignment as constraints and the relevant entities/attributes/relations as variables. By going through a network consistency process, the sources of all the attributes/relations/entities and the referent of each referring expression in the query can be found. The ambiguities are thus resolved (see Chapter 5 and 6).

Two different domains

Source ambiguities are about the uncertainty as to whether the information, i.e., entities, attributes, and relations, originates from the application domain or from the screen display. Therefore, the features of application domains could affect at least some parts of the resolution of source ambiguities.

During our research and in the presentation of this thesis, two different imaginary application domains are used. One is from what we call **non-spatial domains**, and the other is from **spatial domains**. The most important difference between them is that there are spatial relations among domain entities in the spatial domains, but not in the non-spatial domains. This means that if there is a spatial relation mentioned in an interaction involving the non-spatial domain, this relation must have the screen source, while there is no such certainty in an interaction involving the spatial domain.

However, this does not imply that the spatial relations among domain entities are the most important features in resolving source ambiguities. We just notice this difference and use both domains during the course of our research.

The example system from the non-spatial domain is the car selection system mentioned in Section §1.2.1. The example system from the spatial domain is called the **Room Arrangement System**. It is an interactive system for helping users to try various layouts of a room, which, for example, can be useful when they plan to re-furnish their homes. An example is given in Figure 1.4. The system displays a plan (aerial view) of a room, and the icons represent pieces of furniture in the room. How different types of icons depict different types of furniture is shown in the keys to the right of the screen

display: *the colour of the icons represent the type of the furniture; the shapes of the icons are the same as those of the furniture; and the short lines inside the icons mark the front side of the wardrobes and the number of doors it has.* The user can arrange the layout of the room by moving the icons of the furniture in the room, and the screen display will change to show a revised arrangement. During the interaction, the user can also ask for information about the room and the furnishings (see (1.2) for example).

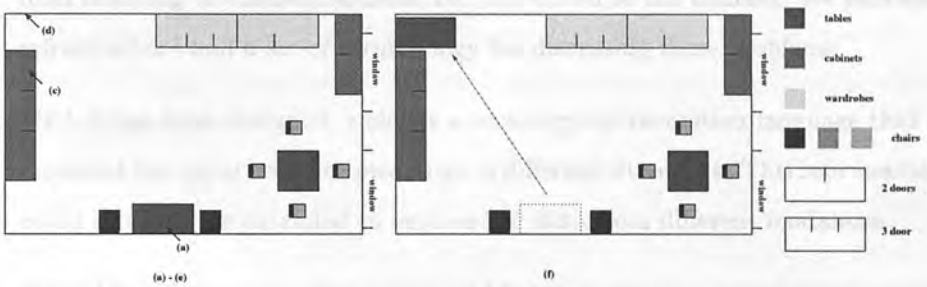


Figure 1.4: Screen displays for (1.2) of the Room Arrangement System

(1.2) **User:** What is the size of this $\nwarrow_{(a)}$ table? (a)
System: `33cm x 66cm.` (b)
User: What is the distance between this $\nwarrow_{(c)}$ cabinet and the wall above it? (c)
System: `You mean this` $\nwarrow_{(d)}$ `wall? The distance is 48cm.` (d)
User: Move the table to the place above the cabinet. (e)
System: `The table has been moved.` (f)

Both systems will be mentioned many times in the following chapters, especially in Chapter 7, which is about evaluation.

Evaluation

Through the above steps, we obtained a computational model for resolving the source ambiguities. In order to verify the function of the model, we ran an evaluation involving human subjects. The evaluation was based on the 'overhearer' method, developed for our specific situation. The subjects acted as overhearer in the dialogues, and gave their verdicts based on their understanding of the naturalness of the dialogues. The evaluation results demonstrate that the model worked, at least for the selected dialogues in the experiment (more in Chapter 7).

1.3.3 Achievements

The achievements of this project are:

1. Some nontrivial reference problems in intelligent multimodal systems have been clarified and identified as source ambiguities, which are the ambiguities arisen from referring to different sources, i.e., the screen or the domain. We provide an infrastructure and a set of terminology for discussing these problems.

2. MRL_S has been designed, which is a meaning representation language that can represent the input sentence meanings in different situations. This representation could probably be extended to express the data from different modalities.

3. Several linguistic regularities within and between referring expressions have been discovered, which are used as heuristics in the resolution.

4. A computational model has been constructed, which uses a constraint satisfaction approach, to resolve singular referent referring expressions with source ambiguities. The phrase types that can be processed are: singular pronoun "`it`", quantified phrases, and definite noun phrases excluding "one" anaphora.

5. A demonstration system called IMIG has been built, which integrates natural language input with graphic display, and the computational model acts as the core. It provides a concrete interface between users and the computer enabling them to fulfil certain tasks.

6. An evaluation of the computational model has been performed, providing some concrete evidences about the advantages and disadvantages of our approach.

7. Overall, this project provides a better understanding of human computer interaction and a better communication environment.

1.4 Organisation of the thesis

The thesis is structured as follows:

In Chapter 2, work in intelligent multimodal research are reviewed. The review addresses the question "whether previous systems can deal with source ambiguities or not", and the result is that a systematic approach to these ambiguities is yet to be developed.

Chapters 3 - 5 describe our three-step approach in detail. Chapter 3 talks about the concepts and the structures that are fundamental to our approach. They include source and source ambiguities, the described entity set, the intended referent of a referring expression, the linguistic regularities and the heuristics based on them, and coreference and quasi-coreference.

In Chapter 4, the meaning representation language, *MRL_S*, which is developed to represent the sentence meanings both before and after the resolution, is presented. MRL_S has two layers: the *QLE* layer is designed for representing the meaning of a sentence with the source ambiguities, and the *LE* layer is for the meaning after all the sources have been clearly identified.

Chapter 5 describes the third step of our approach, a constraint satisfaction based resolution process. It shows how the resolution of source ambiguities can be viewed as a constraint satisfaction problem (CSP), and how the transformation to a representation can be resolved as a CSP. We will also talk about our method of resolving the CSP — a network consistency process — towards the end of this chapter.

Chapter 6 is about the implementation of the IMIG system. As an intelligent multimodal system, IMIG has language processing modules, which realise the computational model we designed. It has various knowledge bases such as the world model, the display model, the mapping model, the context model and the general model.

In order to examine the quality of the computational model, we carried out an evaluation, which is presented in detail in Chapter 7. A discussion based on the results of the evaluation is also provided.

Chapter 8 gives conclusions with some discussion about future work.

Chapter 2

Research on Intelligent Multimodal Analysis

In this chapter, we review previous research on understanding multimodal input. As a quickly expanding research discipline, intelligent multimodal interaction has become an indispensable part of a great number of newly developed systems in the last twenty years. These systems are used in various application domains, and are diverse in terms of system functions and architectures. We do not intend to give a comprehensive review of all aspects of these systems but rather concentrate on aspects that are relevant to our research objectives. These aspects are: advantages of different modalities and multimodal integration, natural language phrases and gestures, knowledge management and multimodal reference processing. In addition to presenting the current state of research on intelligent multimodal interaction in these aspects, the review also aims at identifying the weakness of these systems, which the IMIG system intends to improve upon.

2.1 Overview

As introduced in Chapter 1, this thesis talks about a computational resolution model for processing multimodal queries in an environment integrating natural language text, pointing actions and graphics. These queries could be natural language phrases, pointing actions or both. Because attributes from not only the application domain but

also the visual display could be used in the references, the resolution model must be designed to handle this uncertainty.

The review will concentrate on four aspects that are important to the design of our resolution model. They are *features of modality and multimodal integration, natural language phrases and pointing actions, knowledge management* and *multimodal reference processing.*

Through this review, we wish to present a general picture of the state of art of research in intelligent multimodal systems in relation to the above four aspects and introduce various techniques used in the IMIG system. The main intention is to identify the weaknesses of previous work that the IMIG system intends to improve upon and illustrate the important differences between IMIG and previous systems.

In Section §2.2, we will discuss the advantages and disadvantages of the specific modalities and the benefits of a well designed integration without going into details of the IMIG system.

The remaining three aspects can be paraphrased into the following three questions, which ought to be answered in the development of intelligent multimodal systems. They are presented from the analysis point of view because we are more interested in the interpretation of multimodal input rather than the generation of output.

- How are the input from language, pointing action and graphical modalities integrated?
- How are various types of knowledge organised to facilitate the interpretation?
- How are multimodal referring expressions resolved?

This review is organised around these three questions. In the review, we will mention only the relevant parts of previous work and their weaknesses. A more comprehensive and system oriented review of previous work in intelligent multimodal systems can be found in [Maybury, 1993] and [Maybury and Wahlster, 1998].

2.2 Advantages of different modalities and multimodal integration

Human-human communication is known to be more effective, efficient and natural in comparison with human-computer interaction. Often the human-human communication is multimodal, so it is also interesting to study multimodality in human-computer interaction.

The integration of several modalities is not a simple merging of information from different modalities, but a careful design to achieve function complement, i.e. the advantages of one modality can be used to cover the disadvantages of another in the representation [Maybury, 1993].

The first step towards an integration is to understand the capabilities of each modality. Much research work has been done in this area, amongst which the work of [Cohen et al., 1989] and [André and Rist, 1993, Oviatt et al., 1997] will be presented in the following sections. They are chosen because they are relatively new achievements and are directly related to language, graphics and pointing actions, the modalities we are interested in.

2.2.1 The comparison and integration of language, direct manipulation and graphics

Since 1980's, graphic user interfaces with direct manipulation mechanisms have become the dominant human computer interface mode, in which items and menus on the screen can be directly manipulated by pointing through a mouse or pen.

[Cohen et al., 1989] enumerates the advantages of a "*pure*" natural language system and a "*pure*" direct manipulation system:

- "*Direct manipulation is appropriate when the task is such that a limited number of actions can be taken at a given time, and the objects to which the actions are applied can be made visible on the screen in a coherent fashion.*"
- "*Natural language is particularly appropriate for describing objects and time periods that cannot be referred to directly.*"

- natural language is convenient for expressing quantification information.
- The "*use of pronouns, definite noun phrases and tense allows speakers' utterances to depend on context for their interpretation*", and the use of context makes an interaction much more efficient.

At the same time, they also point out the problems with using only one of the modalities:

- The use of context information in language processing means that the problems of "*resolving anaphoric reference, word sense ambiguities and the attachment ambiguities of prepositional phrases*" may happen in the interaction. All of them are difficult issues in language understanding.
- Natural language systems have another "*often-cited*" weakness — "*the opacity of the system's linguistic and conceptual coverage*". This could make the user over- or under-estimate the ability of a system, which in turn leads to frustration in relation to the user's goals.
- Direct manipulation has difficulties in "*allowing users to apply selected functions to unknown arguments*".
- Direct manipulation also suffers from "*the necessity of using hierarchies of menus that stand between a user and one action he/she wants to perform*". The interaction could be cumbersome if many actions are demanded.

Their conclusion is that a good integrated system should be *synergistic*. The strengths of one modality should be wisely used to cover the weaknesses of another. The complementary interface approach may create a productive relationship among modalities.

Another comparison given in [André and Rist, 1993] is between natural language and graphics:

- Concrete information (such as shape, colour and texture) and events/actions involving physical objects are better presented as graphics than as natural language.

- Spatial information like location, orientation, composition, physical actions and events are better presented as graphics than as natural language.
- Natural language has advantages in expressing temporal overlap, temporal quantification, e.g., `always`, temporal shifts, e.g., `three days later` and spatial or temporal layout to encode sequences, e.g., temporal relations between states, events or actions.
- Semantic relations, e.g., cause/effect, action/result, problem/solution, condition and concession, are better expressed in natural language to avoid ambiguities in picture sequences. Graphics may be used for representing rhetorical relations, such as condition and concession, only if they are accompanied by verbal comment.
- Quantification, especially *most, some, any* and *exactly-n*, is better presented by natural language.
- Negations are better presented as graphics, e.g., overlaid crossing bars unless the scope is ambiguous. In that case, it is better to use natural language.

André and Rist used the above observations in their modality selection in generating multimodal presentations [André et al., 1993]. Their work gives support to the claim that different modalities need to be integrated synergistically to achieve an efficient, effective and natural communication.

2.2.2 Empirical studies about interaction pattern

Both Cohen and André & Rist base their work on intuitions or previous relevant work. Their results are not directly based on empirical studies.

Oviatt and her colleagues did a series of experiments to "*conduct a comprehensive exploratory analysis of multimodal integration and synchronisation patterns*" [Oviatt et al., 1997]. They set up the experimental environment to have people speak and write to a simulated dynamic map system with a pen that is capable of drawing and pointing. Their experiments have three goals:

1. "*to identify when users are most likely to compose their input multimodally, rather than unimodally*".

2. "*to analyse the main linguistic features of multimodal construction, as well as difference from standard unimodal ones*". Their term *multimodal construction* means a multimodal phrase, but it can also be a command.

3. "*to investigate how spoken and written mode are naturally integrated and synchronised during multimodal constructions*".

Among the results reported in [Oviatt et al., 1997], the parts related to our research are:

- In an interaction involving spatial relations, such as the map tasks they tested, users have a strong preference for interacting multimodally. "*100% used multimodal input, and 95% said they prefer multimodal interaction*".

- Not every command sent to the system is multimodal, although users overwhelmingly favour multimodal interactions. The expressions are most likely to be multimodal when the task involves spatial locations, such as *calculate distance, modify, move,* and *add.* These cover 86% of users' sentences involving multimodal constructions (Figure 2.1). Commands that select a specific object displayed on the map (e.g. *zoom, label, delete* and *query*) have an intermediate likelihood of being multimodal. They account for 11% of users' multimodal constructions. The remaining commands in Figure 2.1 are general action commands, which require neither a spatial description nor identification of an in-view object. They are rarely expressed multimodally.

- Multimodal constructions are briefer and less complex than unimodal spoken sentences in the syntactic aspect.

- Speech input is expected to follow the use of the pen within a given lag. The delay between the end of a pen signal and the start of a speech is 1.4 second on average. This delay sounds a bit too long, at least for some researchers in speech processing. Fortunately, we concentrate on text input systems, so the delay would not affect our current work.

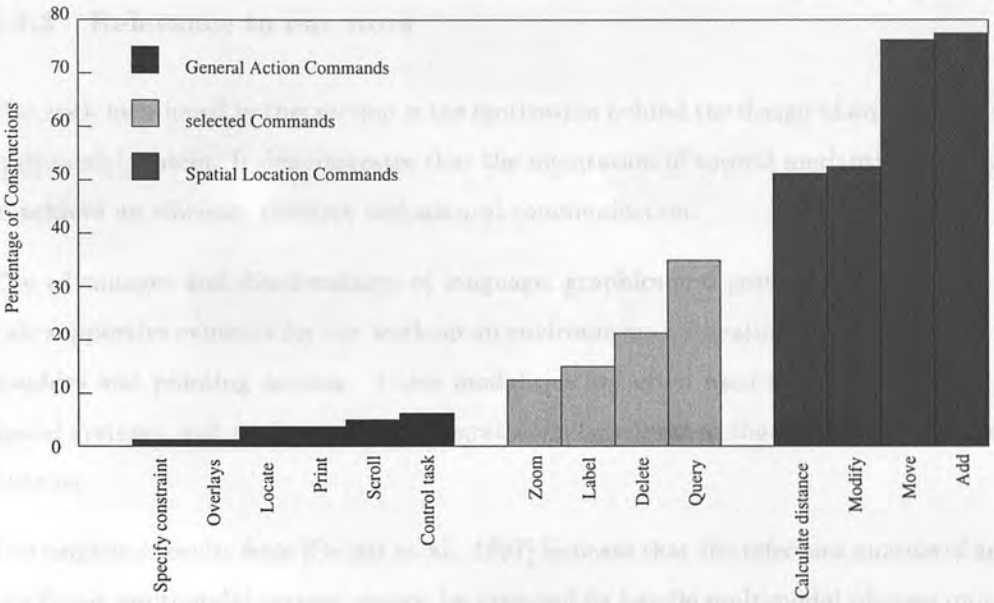


Figure 2.1: The percentage of all constructions that users express multimodally (Extracted from [Oviatt et al., 1997])

- Their results about multimodal constructions (see below) are different from what were assumed by previous multimodal designs, which means that the previous specialised approaches may have limited practical utility. The results are:

  1. most multimodal constructions do not contain any spoken deictic word, whereas previous approaches rely on linguistic flags like deictic words to initiate a disambiguation process for resolving multimodal constructions;
  2. only 25% of multimodal constructions that contain a spoken deictic word need to disambiguate their meanings;
  3. only 17% of multimodal constructions involve a simple point-and-speak pattern.
  4. there are a large number of unimodal sentences.

- "*at a semantic level, the spoken and written modes consistently contribute different and complementary information. Basic constituents describing the subject, verb and object almost always were spoken, whereas constituents describing locative information invariably were written*". Here, the written mode means drawing on the screen using the pen.

2.2.3 Relevance to our work

The work mentioned in this section is the motivation behind the design of an intelligent multimodal system. It demonstrates that the integration of several modalities is a key to achieve an efficient, effective and natural communication.

The advantages and disadvantages of language, graphics and gesture modalities provide supportive evidence for our work on an environment integrating natural language, graphics and pointing actions. These modalities are often used in intelligent multimodal systems, and exploring their integration is beneficial to the development of the systems.

The empirical results from [Oviatt et al., 1997] indicate that the reference module of an intelligent multimodal system cannot be assumed to handle multimodal phrases only. On the contrary, it probably has to process unimodal referring expressions most of the time. For this reason, the reference model of the IMIG system has the ability to handle not only multimodal phrases, but also common unimodal phrases. In addition, because source ambiguities can appear in both multimodal phrases and unimodal phrases (see Chapter 3), the reference model of the IMIG system must have the ability to handle both cases.

2.3 Natural language phrases and gestures

Natural language and gesture are two modalities often used in intelligent multimodal systems. To achieve a complementary combination of two modalities, many systems have developed their own strategies.

All multimodal systems have a single internal representation for the information from modalities, which acts as a message carrier to indicate what has entered by the user and what the system will generate. In fact, some people view this common representation as the most important feature that distinguishes a multimodal system from a multimedia system [Wilson et al., 1991]. For example, Figure 2.2 shows a common representation connecting language, graphics and gesture(pointing) modalities.

The discussion in this section concentrates on three issues: 1) the methods used for

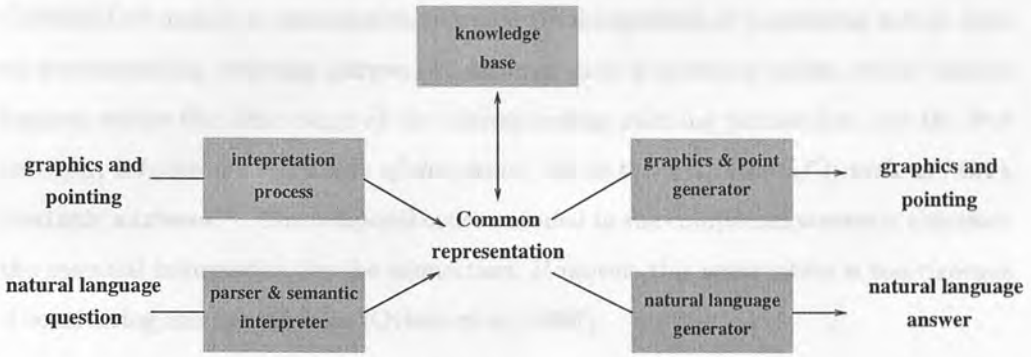


Figure 2.2: Common representation in a multimodal interaction integrating natural language, pointing actions and graphics

analysing input from the language and gesture modalities, 2) the common representation to which the input is transformed and 3) the connection mechanisms between pointing actions and their corresponding phrases in the language input. The review in this section aims to provide background information in the literature and to indicate why we choose our method to implement the processing of pointing actions in the IMIG system.

2.3.1 The CUBRICON system

The CUBRICON system was developed in the late 1980's and was applied to the domain of military tactical air control [Neal et al., 1988, Neal and Shapiro, 1991]. The system receives input from three devices: speech, keyboard and mouse. It renders its output to three other devices: a colour graphic display for map and text, a monochrome display for table and a speech device for short natural language text.

Viewing graphics, natural language and pointing as different parts of the same language, which is called the "*unified view of language*" [Neal and Shapiro, 1991, p. 19], the CUBRICON system combines input from the three modalities into a single compound stream, but at the same time maintains the original temporal order of the tokens. The compound stream is analysed by an extended ATN based parser to produce an interpretation. In this sense, the compound stream is the common representation of the CUBRICON system.

CUBRICON makes an assumption to guide the integration of a pointing action with its corresponding pointing phrase. It assumes that a pointing action would always happen within the time range of the corresponding pointing phrase (i.e. "*as the first token(s), between the NL words of the phrase, or as the last token(s)*"), such as "`this <point> airbase`"[1]. The temporal order encoded in the compound stream is therefore the essential information for the connection. However, this assumption is too rigorous if considering the results from [Oviatt et al., 1997].

2.3.2 The XTRA system

The XTRA (eXpert TRAnslator) system, which was developed by Alfred Kobsa and his colleagues at the University of Saarbrücken, adds an intelligent multimodal interface to an expert system to assist users to fill in tax forms. It integrates natural language, graphics and pointing in both input and output [Allgayer et al., 1989, Wahlster, 1991]. The parser is based on SB-PATR, a unification-based parser for German. All its knowledge is encoded into SB-ONE, a KL-ONE-like language [Brachman and Schmolze, 1985], in which a knowledge base called *functional-semantic structure (FSS)* stores the representation of the input sentence after syntactic and semantic analysis.

Through mouse and on-screen animated image display, XTRA supports the following four pointing modes, which are necessary because of the combination of accurate and vague points in their table application domain.

1. exact pointing with a pencil,
2. standard pointing with an index finger,
3. vague pointing with an entire hand, and
4. encircling regions with the '@'-sign.

To achieve the four mode pointing, the XTRA system uses two steps to perform a pointing action. Firstly, one pointing mode is selected, and then the intended area or entity on the screen is pointed at.

[1] "`<point>`" is their notation for the pointing action accompanying a phrase and is not a part of the linguistic input. It has the same function as our sign ↖ .

XTRA does not combine natural language input and pointing gestures into one compound stream for processing, as in the case of the CUBRICON system. Instead, it handles them in different modules. The language input is morphologically and syntactically analysed in two modules called MORPHIX and SB-PATR respectively, and the deictic input is processed in the TACTILUS module, a component specialised in handling tasks such as filling tables. These two pieces of information will be combined in a later stage of interpreting referring expressions (see Section §2.5.2).

However, XTRA uses the same assumption for the connection between pointing actions and pointing phrases as the CUBRICON system. It assumes that a pointing action would happen in the range of the corresponding pointing phrase, i.e., either just before, in the middle, or just after the pointing phrase [Schmauks and Reithinger, 1988, Allgayer et al., 1989]. They acknowledged the lack of empirical evidence for the assumption, and used the rule based on the assumption only as a heuristic [Schmauks, 1987].

2.3.3 The ALFresco system

The ALFresco system is an interactive system for users who are interested in fourteenth century Italian frescoes and monuments [Stock and Team, 1993, Carenini et al., 1993]. Its major concern is the integration of natural language with hypermedia to facilitate the organisation of heterogeneous and unstructured information and to remove the disorientation and cognitive overhead problems that could happen in hypermedia navigation. The system is connected to a video disc unit and a touch screen.

ALFresco receives input from a keyboard and a touch screen, and renders prompt answers and more complex descriptions through natural language or images. The touch screen provides a more natural pointing mechanism than the mouse, but it suffers from low accuracy.

The linguistic input is analysed by a chart-based parser called WEDNESDAY 2. Going through the semantic analysis, the input is translated into an expression of World Model Language, and then dynamically mapped into the YAK assertional language. YAK is a language ALFresco uses to build its knowledge base [Franconi, 1990]. The language is a descendant of KL-ONE.

It is not clear how pointing action input is processed in the ALFresco system, and there is only a very brief discussion about the mechanism for connecting a pointing action with the corresponding phrase in the descriptions of the system. The assumption is that the user would point to the screen immediately after he/she types in a demonstrative, that is, "*once the user has typed a demonstrative, the system admits a possible pointing gesture on the touch screen, until the user types further words*"[Carenini et al., 1993]. Compared to the assumption used in CUBRICON and XTRA, the approach in ALFresco demands even more co-operation from the user.

2.3.4 The EDWARD system

The EDWARD system was developed to explore the so called *fully integration* of several interface styles [Bos et al., 1994, Huls et al., 1995], where natural language (Dutch), manipulation of graphical representations, menus and command language can all be used at any time during an interaction. This could be seen as an extension to the abilities of the above three systems, where the language mode is always the primary means of interaction.

The EDWARD system accepts user input from two devices, the keyboard and the mouse, and generates output to either the text window or the graphics window. It integrates a graphical graph-editor called GR^2 and a Dutch natural language dialogue system called *DoNaLd*, so it is reasonable to assume that the gesture input and the language input are processed separately in the two subsystems before they are merged for referent resolution.

Concerning the issue of connecting pointing actions to the corresponding phrases, the EDWARD system also uses the temporal relation as the essential information. It assumes that the pointing gestures would be simultaneously co-presented with the typing of the demonstrative of a noun phrase, which is based on the experiment of [Marslen-Wilson et al., 1982] on human verbal conversation and pointing gestures. However, since the language input to EDWARD is in written form rather than verbal form (as is the case in [Marslen-Wilson et al., 1982]), the applicability of Marslen-Wilson et al's result is questionable.

2.3.5 The MMI^2 system

The MMI^2 system is a toolkit connected to a knowledge based system. It provides multimodal interfaces and undertakes co-operative dialogue with the user [Wilson et al., 1991, Binot et al., 1992, Ben Amara et al., 1991]. The MMI^2 system uses a single internal representation language for information in all modes, which is a strong commitment to being "*multimodal*" (as one of its authors says [Wilson et al., 1991]). The system supports various forms of input from different modalities, such as natural language, command language, graphical display, direct manipulation and gesture. It also produces graphical visualisation for viewing the state of its knowledge bases.

The MMI^2 system processes input from text and gesture in different modules. The *graphics manager* handles gesture input, and the command language/natural language module handles input in command language or natural language form. The processing results are combined into a logical expression called *common meaning representation*, which is the common representation within the MMI^2 system.

No detail about the issue of connecting a pointing gesture with its corresponding phrase in MMI^2 is given. However, from the examples used in the discussion of the system, it seems that the temporal relation is still the key. [Binot et al., 1992] indicated that the corresponding pointing gesture would happen just before entering to the first word of a pointing phrase. As we pointed out, the appropriateness of this assumption is questionable.

2.3.6 Hayes' work

Although his work was a few years earlier than any of the above systems, Hayes actually did more thorough work on integrating natural language phrases and pointing gestures [Hayes, 1986].

Hayes' work is not about a fully implemented intelligent multimodal system, but rather a discussion of some issues in building such a system. In [Hayes, 1986], he called the intermixing of pointing gestures and natural language input *natural language pointing*. Hayes indicates that the use of natural language pointing is not straightforward because of the following three questions:

- How to "*determine when pointing events are natural language pointing events*"?
- How to "*determine where the entities pointed at fit within the overall interpretation of the natural language input*"?
- How to "*identify which entity was actually pointed to (an issue when the visual representations of entities are nested within each other on the screen)*"?

Hayes proposed several potential solutions to the first question, but none of them were ideal. He chose the following, which is also used by most of the previous systems, as the best:

> "*Assume that all pointing events during (or close to) natural language input are natural language pointing events. This has the advantages of simplicity, robustness and lack of ambiguity. Its main disadvantage is making world navigation impossible during natural language input (even during a pause for thought during type in).* "

The second question is about connecting pointing actions with the corresponding phrases. Hayes found that, although there could be several possibilities relating to the co-occurrence of words and pointing actions, "*the only real invariant seemed to be that the natural language pointing events will occur in the same order as the phrases in the natural language input to which they correspond.*" In addition, phrases with a demonstrative determiner are more likely to correspond to natural language pointing than those with a definite determiner.

Hayes gave a heuristic procedure for connecting pointings and phrases.

"1. *list in left to right order all schemas representing phrases in the input that could potentially correspond to natural language pointing events;*

2. *list in left to right order all natural language pointing events;*

3. *form all lists of pairings between schemas and pointing events, such that:*

   (a) *the left to right ordering is preserved for both schemas and pointing events;*

(b) paired schemas and pointing events are compatible in some way;

4. *if more than one list of pairings remains, choose those with the maximum number of schemas that correspond to phrases that are linguistically likely to be deictic (including all pronouns and noun phrases with demonstrative determiners);*

5. *if more than one list of pairings still remains, choose those with the shortest time mismatch between the pointing events and the phrases corresponding to the schemas;*

6. *if there is still more than one, ask the user to decide."*

Obviously, Hayes chose a much weaker assumption about the synchronisation between pointing actions and the corresponding phrases. His algorithm is more elegant than that of any of the above systems. However, as he admitted, there is no conclusive evidence that his solution would perform correctly in all situations. Therefore, he treats his solutions as heuristics, which are open to empirical testing.

Unfortunately, his discussion about the third question is vague and less detailed, so we omit it from this review.

2.3.7 Relevance to our work

We have introduced the input analysis of several intelligent multimodal systems, and in particular their methods for dealing with natural language phrases and pointing gestures. Here we summarise the current state of research on this topic:

- Most systems process the input from language and gesture modalities separately, except for the CUBRICON system. The separate processing methods are chosen for their simplicity, although there could be cases where the information in different modalities is helpful to each other.

- All systems mentioned are connected with an expert system or some other knowledge based system. Multimodal communication shows its strength in knowledge rich interactions.

- Most systems integrate natural language, graphics and pointing actions to facilitate communication between the user and the system. All systems use the keyboard as the main or sole input device for natural language and a mouse as the only pointing device, except for the ALFresco system which uses a touch screen. In the systems using a mouse, there are one-granularity and multi-granularity pointing gestures. For example, the XTRA system supports four different types of pointing, while CUBRICON only supports one.

- To connect pointing gestures with the corresponding phrases, most systems use the temporal relations between the pointing and the phrases as the key information. To deal with multiple pointing gestures and multiple phrases in one sentence, many systems restrict the pointing actions to happen within the time range of the phrases, or to be immediately after the entering of the demonstratives. However, since Oviatt et al. have pointed out that there is no such regularity in human generation of phrases and gestures (see Section §2.2.2), this method may not work very well in a practical situation.

The IMIG system can make use of the following ideas and extend them at the same time:

- It is reasonable that the IMIG system integrates natural language (keyboard entered text), graphics and pointing through a mouse.

- The fully integrated multimodality of the EDWARD system is a good way of extending the ability of a system. The IMIG system follows this idea and allows the user to choose any modality or modality combination during the interaction, including pure screen operations.

- The IMIG system uses Hayes' heuristic algorithm to connect pointing actions with their corresponding phrases. The algorithm is fairly straightforward, and its constraints on the user's behaviour are less restrictive than that in other systems. Furthermore, the algorithm remains valid even when considering the results of Oviatt et al.

2.4 Knowledge Management

We mentioned that most intelligent multimodal systems are used in knowledge rich application domains. To facilitate its communication with the user, each system has to organise its knowledge in some way.

Most intelligent multimodal systems have three types of knowledge, knowledge about the world and the application domain, knowledge about the visual display and knowledge about the context of a dialogue. However, the strategies used to manage these types of knowledge vary from system to system. In the following, we will review some knowledge management strategies used in these systems.

2.4.1 The CUBRICON system

The CUBRICON system uses two separate knowledge bases for domain independent knowledge and domain-specific knowledge that are only applicable to the air-base control domain. It also has another database called the *discourse model*, which contains not only the context information represented as a main focus list, but also the visual display information stored in a sub-module called the *display model*. This organisation is due to the idea behind CUBRICON that objects presented on the graphic displays are intentionally "*expressed*" or "*mentioned*".

The main focus list contains the entities explicitly expressed (either by the user or by the system) via natural language, pointing, highlighting or blinking in previous dialogues. It uses the focus space theory of Grosz and Sidner [Grosz and Sidner, 1986] to organise the entities in the list.

The display model stores all objects that are visible on one of the monitors. Here the objects are domain entities. Therefore, in CUBRICON, except for the fact that their icons are shown on the screen, the entities in the display model are the same as those in the main focus list in the sense that they are both from the domain. The icons in CUBRICON are merely taken as descriptions. Some visual display features, such as highlighting, blinking etc. are treated as extra information about the domain entities.

Although both types of information are stored in the discourse model, the entities in

the main focus list and those in the display model are separated conceptually. This can be illustrated by the fact that entities in the display model are less salient than those in the main focus list. Only when an appropriate referent for a definite referring expression is not found in the main focus list, is the display model consulted.

The system handles the display model in a similar manner to the main focus list. The display model is just a set of entities from the domain-specific knowledge base. This view of the display model is different from ours (see Chapter 1 and 6).

2.4.2 The XTRA system

In the XTRA system, the world and domain-specific knowledge is stored in a SB-ONE like knowledge base called the *conceptual knowledge base (CKB)*. As a semantic network, CKB maintains knowledge of a universal nature at the general level and knowledge about individual facts and objects in the domain of discourse at the individual level.

In addition to handling deictic input as mentioned in Section §2.3, TACTILUS has the functions like "*the interactive construction of forms on the terminal screen, the internal representation of their structure, and for the analysis of pointing to the displayed forms*" [Allgayer et al., 1989, p. 167]. In a simplified sense, it is the place for storing the visual information of the XTRA system. Probably due to the domain characteristics of having various slots and values for both domain and screen entities, the system does not discriminate domain entities/features from screen entities/features.

XTRA stores the interaction context in a module called the *linguistic dialogue memory*, where the referential objects, dialogue sequence memory and dialogue context model are stored. The salience information is encoded as dynamic focus values of each referential object in the dialogue sequence memory. Unfortunately, there is no convincing evidence for the computation of those values.

Similar to the CUBRICON system, XTRA automatically adds visible objects and events on the screen into the dialogue focus, because it believes that the tax form displayed on the screen should meet the *physical co-presence* condition of [Clark and Marshall, 1981].

The use of the visual display knowledge base in the XTRA system is the same as that in the CUBRICON system, which is different from our view of the display model.

2.4.3 The ALFresco system

The ALFresco system expresses the domain knowledge in YAK, a descendant of KL-ONE. As a hybrid system. YAK's terminological component (T-box) consists of a tangled hierarchy where generic concepts and attributes (roles) are defined, and its assertion component (A-box) consists of instances that are presented as frames connected to T-box.

The organisation of the context and visual information in the ALFresco system is similar to that of CUBRICON and XTRA. Logically, there are two components for dealing with the context and visual information respectively.

Also with the view that entities on the screen should be in the context, ALFresco puts both context and visual information in a database called the *topic module* [Samek-Lodovici and Strapparava, 1990, Zancanaro et al., 1997]. The design of the topic module integrates global focus strategies [Grosz and Sidner, 1986], local focus approaches [Hajicova, 1987] and deictic reference techniques [Wahlster, 1991].

The topic module has a component called *deictic context* for storing the visual information. The deictic context changes whenever a new fresco is displayed on the screen. Accompanying the appearance of a new fresco, the pre-indexed characters, e.g., persons, saints, angels and animals, and the holy scenes, e.g., annunciation and resurrection, are added into the deictic context. The ALFresco system does not discriminate domain entities/features from entities/features on the screen either.

The topic module also handles the linguistic context, and it represents the attentional focus based on the centering theory [Grosz et al., 1995]. To extend the centering theory to target dialogues, the ALFresco system defines the minimal unit to be a **turn**, i.e., an adjacent pair of question/answer or request/response, rather than an utterance [Zancanaro et al., 1997]. Consequently, other parts of the centering theory are adopted accordingly, e.g, the backward looking center $C_b(T_n)$ and forward looking center $C_f(T_n)$ are for turns rather than utterances.

2.4.4 The EDWARD system

The domain knowledge of the EDWARD system is represented as a semantic network, which is implemented in CommonORBIT [De Smedt, 1987], a frame-based language similar to KL-ONE.

In contrast to the above systems, EDWARD does not organise the context and visual information into two logically separate components. Based on the framework of [Alshawi, 1987], EDWARD integrates the linguistic and non-linguistic information in dialogues into one database called the *context model* [Huls et al., 1995]. The reason for this integration is that "*one mechanism that handles both deictic and anaphoric expressions in the same way is preferable*" from the computational and engineering point of view.

The central notion in their context model is **salience**. The salience of an individual entity is determined by various factors with different importance, e.g. the recency of the last mention, the syntactic roles, the visibility (because the system involves graphic display) and the so called *gradedness*, i.e., entities in the model gradually become less salient [Huls et al., 1995].

The central construct of the model is the *context factor*, which is defined by a scope (i.e. a collection of individual instances), a significance weight and a decay function. Different weights and decay functions are assigned to entities mentioned by linguistic means (such as referents) and entities that are visible, selected or indicated.

Same as the above three systems, EDWARD treats entities on the screen as in the context. So it enumerates all entities on the screen in the context model. Screen entities have lower salience than those mentioned through linguistic means, and their salience values stay the same until they are either removed from the screen or are pointed to (in this case they become the most salient entities in the model).

2.4.5 MMI^2

Different from the pipeline architecture used by all the above systems, the MMI^2 system has a star like structure. The major control unit at the centre of the star is called *the*

dialogue controller, whose function is anaphora and deixis resolution. Other modules are called *experts*, which are specialised in particular aspects of dialogue management. One of them is the *dialogue context expert*, which manages the focus of the dialogue. Another is the *graphics manager*, which represents and organises everything in the graphics format. Therefore, the reference module, context model and display model are achieved by these three components respectively.

The context expert simply stores "*every interaction ... as it comes from or its sent to the different modes*" [Binot et al., 1992]. This is to save time on unused retrieval and to make the system more practical and flexible for software design.

Although problems similar to source ambiguities are mentioned vaguely in one of their papers [Binot et al., 1992], their graphics manager still organises the representation of screen entities in a way analogous to the context model. That is, there is a list of possible discourse referents that are visible on the screen, which can be updated by the graphics manager. However, they do not mention whether, and if so where, descriptions of the graphical representation of the domain objects, i.e., icons, are stored.

2.4.6 Relevance to our work

Although all the above systems view entities on the screen as "*mentioned*" in the context, two types of methods are used to organise context and visual display information. The EDWARD system uses only one context model to store both types of information, and genuinely combines them together in language processing, whereas the other systems keep the two types of information separate, at least in the logical sense. However, we do not think entities on the screen should be automatically put into the context. Although entities on the screen have a higher chance to be referred to than those neither in the context nor on the screen, it is still questionable to mix screen entities with entities in the context. At least screen entities cannot be forgotten, whereas those in the context can. Our IMIG system does not automatically add entities on the screen into the context, and uses two separate databases to store entities in the context and those on the screen.

In addition, to resolve source ambiguities, we think that the database storing entities

on the screen (i.e. the display model) should be similar to the database storing domain entities (i.e. the world model) rather than to the context model. This is based on our belief that the elements of the display model should not be the domain entities whose icons are on the screen, but rather the icons themselves. It is the descriptions of icons that are needed for resolving source ambiguities. In this sense, icons should be treated as stand-alone objects whose own features are shown on the screen. The display model is the place for storing icons and their features, just as the world model is a place for storing the domain entities and their features. Through some kind of mapping relations, screen entities are connected with the domain entities, and these mapping relations are also critical for the resolution of source ambiguities. Therefore, the IMIG system has a display model, a world model and a mapping model between the two models (see Chapters 3 and 6 for details).

In the reviewed systems, the knowledge bases related to referent resolution are all accessible to the reference model. The IMIG system uses the same idea. In addition, the reference model of IMIG can retrieve visual attributes from the display model and the mapping relations from the mapping model. This does not exist in all the reviewed systems.

Similar to the other systems, the IMIG system has a common meaning representation language, which encodes detailed information about the sources of various components of the input. This information can be used for source ambiguity resolution.

The EDWARD system uses Alshawi's framework to model the context using salience factors. Although this method is probably preferred from a computational and engineering perspective, it has various limitations. For example, the numerical representation of salience sounds ad hoc, and several discourse phenomena mentioned in the literature, e.g., the discourse intentions, the discourse segmentation and the different context effects of a pronoun and a definite NP, are not considered in EDWARD's context model.

In fact, more problems could arise when the EDWARD system is used in an application domain that needs more complex screen displays. Its current application domain is a file system, and the on-screen display is relatively simple compared with most of the

other reviewed systems. When there is a demand for a more complex screen display and a more detailed representation of the visual information, their context model needs to provide more support for a wide range of visual representation rather than just a list of objects on the screen. Consequently, the visual representation may become too complex and too large to be fully integrated into a dynamic component like the context model.

Despite these, the IMIG system borrows the following ideas from previous work:

- Entities on the screen could be referred to in subsequent dialogues, although this is more likely to be the case for those entities in the context.
- As a dialogue system, the IMIG system follows the ALFresco system and organises its context information based on a turn of dialogue rather than a sentence.

2.5 Multimodal reference processing

Multimodal reference processing relates to how the input referring expressions, possibly accompanied by pointing actions, are processed in intelligent multimodal systems. It is interesting because both linguistic context and information from other modalities are available for interpreting referring expressions. Therefore, the reference model of an intelligent multimodal system has to be more powerful to handle both of them.

Most previous work concentrates on handling information from linguistic sources and pointing actions. Little work touches the issue of using visual display attributes in referring expressions and handling phrases referring to entities on the screen. They are the exact causes of source ambiguities.

The following sections review the work related to multimodal reference processing in detail. The discussion centres around the issue of whether or not the reference model of each reviewed system has the abilities to process the following:

- expressions with pointing actions.
- expressions with visual display attributes.

- expressions whose referents are entities on the screen.

2.5.1 The CUBRICON system

The CUBRICON system can handle expressions with pointing actions but not the other two types above. It uses an ATN based parser to find referents for the input referring expressions. To be more precise, when a phrase accompanied by pointing actions is parsed, the parser uses the corresponding point coordinates to determine which icons on the screen are touched. A list of potential candidates for the referent is generated. Restrictions from linguistic and other world knowledge are then imposed on those candidates. If the number of candidates satisfying the restrictions is equal to the required number, these candidates are the referents of the phrase. Otherwise, the system needs a disambiguation or relaxation process to find the referents.

The system "*accepts and understands multi-media input such that references to entities in a natural language sentence can be accompanied by co-ordinated simultaneous pointing to the respective entities on a graphics display.*" It "*is able to use a simultaneous pointing reference and natural language reference to disambiguate one another when appropriate. It also infers the intended referent of a point gesture that is inconsistent with the accompanying natural language*"[Neal and Shapiro, 1991, p. 12]).

The inconsistency between a phrase and the accompanied pointing gesture means that: 1) the pointing gesture touches at least one object, but the touched object(s) does not match the description of the phrase; or 2) what the user points to is a screen location that contains no object. CUBRICON uses a process called "*bounded incremental spatial(geographical) search*" to find the nearest object(s) that satisfies the filtering criteria. The filtering criteria are generated from the knowledge resources used in the resolution process. An error message would be displayed if the attempt to resolve the inconsistency fails.

This system probably does not have the ability to deal with referring expressions with visual display attributes or screen referents because it does not store other information about the visual properties of the graphic objects except for a list of entities in the display model. We believe that the existing visual display information in the system

is far from sufficient for such a capability.

2.5.2 The XTRA system

Similar to CUBRICON, the XTRA system also concentrates on resolving referring expressions with pointing actions. Referent identification is performed in the interpretation module, which can access results from linguistic and deictic analysis for processing multimodal phrases.

In contrast to the CUBRICON system, which always uses the result from the deictic analysis as the basis of the linguistic restrictions, XTRA adopts a more flexible approach. Both linguistic and deictic analysis can deliver a set of referential candidates for a single description under certain conditions. XTRA uses heuristics to judge which analysis results should be used first [Kobsa et al., 1986]. This approach has the benefit of reducing computational complexity, but it may impose restrictions on the portability of the system, since the heuristics may vary in different domains.

The referent identification process in the XTRA system has the following three steps:

1. Potential referents are generated by either linguistic analysis or deictic analysis according to the heuristics.
2. Each candidate is evaluated by consecutively considering information from all other knowledge resources, including case-frame analysis and domain knowledge.
3. The candidates are all evaluated by considering all partial plausibility assignments, and the one with the highest plausibility factor is selected as the intended referent.

Again, we find no discussion about referring expressions with visual display attributes or screen referents. Considering the available knowledge in the XTRA system, it is unlikely that XTRA has the ability to deal with source ambiguities.

2.5.3 The EDWARD system

The reference resolution in the EDWARD system is a search process in the context model [Huls et al., 1995]. It first lists all individual entities satisfying the semantic restrictions of the phrase, and then calculates the salience value of each entity by adding the significance weights of all the context factors corresponding to the entity. The most salient individual entity is taken as the referent of the phrase. Differing from the above systems, the resolution of anaphoric and visual situation use of a phrase are not performed in separate search processes in the EDWARD system.

The EDWARD system divides the phrases it can interpret into three groups. *Unimodal linguistic reference* is one of them, which includes anaphoric phrases (i.e., definite phrases, anaphoric expressions using demonstratives, personal pronouns and adverbs) and deixis (i.e., personal deixis like "`I`" and "`you`" and spatial deixis like "`the closed bookcase`", where there is only one icon which resembles a closed bookcase on the screen, or "`the bottom most file`"). The second group is *unimodal graphical reference*, which actually contains pure screen operations directly manipulated by the user. The third group is *multimodal referring expressions*, which can be the multimodal phrases in the common sense (i.e. whose demonstrative words are accompanied with pointing gestures) as well as non-demonstrative definite NPs as pointing phrases, such as in "`the report about DoNaLD is in Claassen`↖ ".

To test the ability of the reference model, Huls et al. describe an evaluation involving the EDWARD system and two other models. One of them is a simplistic model that only picks up the last mentioned semantically appropriate referent, and the other is a model based on [Grosz and Sidner, 1986]. The evaluation uses 125 real, user-generated referring expressions. The EDWARD system performed the best and found the correct referents for all the phrases (precision 100%). Grosz and Sidner's model missed only one (precision 99.2% (i.e. 124/125)), and was better than the simplistic model, which got 119 right (precision 92.8% (i.e. 116/125)). The evaluation demonstrates the quality of the EDWARD model. However, questions about the appropriateness of the test collection could be inferred from such near perfect performance of all three methods. It is well known that referent evaluation is a difficult problem and no method has achieved

or even approached such high precision in a real practical situation. For example, a recent work by [Vieira, 1997] just reports a 80%-85% precision in the analysis of definite descriptions in unrestricted text. This shows that their test collection might be too simple in comparison to what happens in real situations.

In summary, the two properties of the EDWARD system, i.e., fully integrated multimodality and unified context model, make it better than many other systems. However, it has the following limitations:

1. When the screen display becomes more complicated, which is likely to be the case as the ability of the intelligent multimodal systems improves, it is not clear whether the visual display information can still be reasonably represented in the context model. If a separate visual representation is inevitable, it is not clear what effect it would have on their reference process.

2. The application domain of EDWARD is the graphical display of a file system. Accompanying the graphical display, there are spatial relations between icons of files and directories. Although the use of spatial relations in referring expressions were mentioned in the EDWARD system [Huls et al., 1995], referring expressions with other types of visual attributes or screen referents are beyond the ability of the system. As reviewed in the discussion about its knowledge management, it does not have screen entities and visual attributes (apart from spatial relations) represented in it.

2.5.4 The MMI^2 system

The referent resolution method of the MMI^2 system relies on two principles: 1) there is a common meaning representation for the input query, and 2) the dialogue controller controls all the processes [Binot et al., 1992].

However, the paper does not provide more information about the resolution process of the MMI^2 system. It raises many interesting problems without providing mechanisms to tackle them. The discussion about problems similar to source ambiguities is an example.

In the discussion of the graphics manager, the issue of referring to objects using their graphical features, e.g., "`purple workstation`", is mentioned [Ben Amara et al., 1991]. To handle this type of references, Ben Amara et al. briefly mention that the graphical features of objects should be available to the referent resolution mechanism, like the real world features. This is one of the central arguments of this thesis. Binot et al. also mention the issue that the referent of a phrase could be a graphical icon on the screen, for example, in the sentence "`add a PC to its left`", the referent of the pronoun is a screen icon [Binot et al., 1992]. However, they acknowledge that they do not have the solution to the above problems.

2.5.5 Multimodal generation systems

The COMET system and the WIP system are two intelligent multimodal presentation systems, which provide modality coordination in *generating* multimodal presentations. They are reviewed here because of their mechanisms for handling multimodal referring expressions.

The COMET system

The COMET system was developed to explore the generation of coordinated, interactive multimedia explanations (combining text and three-dimensional graphics) of equipment maintenance and repair procedures [Feiner and McKeown, 1990a], [Feiner and McKeown, 1990b, Feiner and McKeown, 1993, McKeown et al., 1992].

The generation process starts with content planning, which produces a hierarchy of logical forms, the common descriptions of both the text and graphics to be generated. The logical forms are annotated with directives indicating which information is to be realized by which media in a module called media coordinator. The result from the media coordinator goes through two parallel processes, text generation by a module called the text generator and graphic generation by a module called the graphic generator. Referring expressions are realised in the text generator, based on the content decided by the content planner or the media coordinator.

The graphic generator maintains a representation of the illustrations on the screen

so that other components of the system can query them. In this sense, the graphic generator contains the display model of the COMET system.

COMET can handle *cross-references*, which is a type of referring expression with part of the expression referring to materials in another modality. COMET can generate two forms of cross-references, *structural cross-references*, which refer to the structure and layout of the illustration, and *content cross-references*, which refer to the content of an illustration. Spatial relations, relative to either the illustration itself or another object in the illustration, and special graphical features of an illustration, e.g., highlighting and cutaway view, are counted as a part of the content of the illustration. Therefore, they can be used in the content cross-references.

Requested by the user input or the nature of the current presentation, i.e., the system assumes that the user does not know how to describe an object, a cross reference can be generated by a sub-module in the media coordinator called the *cross reference generator*. During cross reference planning, the cross-reference generator queries the graphic generator for information about the special graphical features (including visual effects or layout), visibility and so on.

The COMET system treats the graphics on the screen as descriptions of the domain objects in the current context. Although the system can generate phrases like "`the highlighted mode knob`" where "`the highlighted mode`" is a visual display attribute, it cannot produce phrases like "`the yellow icon`" even if yellow is the way of highlighting and the knob is an icon on the screen. This is due to it not treating graphics as individual entities and therefore not having a yellow icon in its knowledge base. This might not be a problem for generation unless the system has no other means of referring to an object. However, this would be a limitation to an intelligent multimodal interface if the user uses phrases which the system cannot interpret. We have argued in Chapter 1 that the user might use such phrases.

The WIP system

The WIP system is also dedicated to the coordinative presentation of information in different modalities [André et al., 1993, André and Rist, 1995]. The major contribu-

tion of this system is to apply AI and NLP techniques, such as planning, semantic and pragmatic processing (coherence, anaphora, rhetorical relations, speech act), to the multimodal domain. We will mainly consider their work on anaphora.

The generation process of the WIP system starts in the *presentation planner*, which works in coordination with the *layout manager* to generate a description of the presentation plan. This process includes determining the contents of to be generated information and selecting appropriate mode combinations. As in COMET, the result then goes through two parallel processes, which deal with the design and realisation of text and graphic output respectively.

In the design and realisation of output, the components of a specific modality can communicate with the presentation planner about any detail or revision of the presentation. It is reasonable to suggest that there is a place to store the details of the graphics to be generated so that the text output can query it. In some sense, this place has the function of the display model.

In the WIP system, the cross reference phenomena are called *cross-media references*, which are defined as phrases that "*do not refer to world objects, but to document parts in other presentation media*" [Wahlster et al., 1991]. They provide a model to explain the coreferential links between referring expressions, objects of the world and objects of the multimedia presentation [André and Rist, 1994]. They aim at providing a representation that the user can use to link his/her mental representation to the object being described. According to them, there are four types of cross-media references:

1. linguistic anaphora with linguistic antecedent
2. linguistic anaphora with pictorial antecedent
3. pictorial anaphora with pictorial antecedent
4. pictorial anaphora with linguistic antecedent

They observed that graphical attributes of objects could be used in linguistic anaphora, for example, "`the dark switch`" where **dark** is a graphic attribute. This indicates that their model can handle the visual display information in a phrase. However, as

in COMET, the WIP system treats pictures on the screen as descriptions of domain entities. Therefore, it does not have a systematic representation and resolution mechanisms for phrases whose referents are the graphic objects and those whose descriptive contents are visual display attributes of graphic objects.

2.5.6 Relevance to our work

While some systems use their reference models to deal with the interpretation of referring expressions with pointing actions, others (such as MMI^2, COMET and WIP) extend the ability of their models to handle more difficult issues such as phrases with a few visual display attributes. Our IMIG system focuses on source ambiguities and tries to resolve not only phrases with a few visual display attributes, but also phrases composed of only visual display attributes and phrases whose referents are screen entities. It aims at providing a systematic method for resolving source ambiguities.

In the reviewed systems, the mechanisms used for multimodal reference processing are closely related to knowledge management. For example, the mechanism in the EDWARD system is different from that of the other systems because EDWARD organises linguistic context and visual information in a different way. This is also the case for the IMIG system, which views the display model as the counterpart of the world model, and has a mapping model between them. All of these three models can be used in the referent resolution process. In addition, because source ambiguities and common referential ambiguities can appear at the same time, the IMIG system uses a constraint satisfaction method to access knowledge in various knowledge bases and resolve both types of ambiguities simultaneously.

Although André & Rist mention problems similar to source ambiguities, their assumptions about the user's mental state are different from those of the IMIG system. Their model concerns the generation of cross-media references, and it assumes that the user already knows the existence of an object, but is unable to locate it on the screen and in the application domain. In contrast, the IMIG system mainly concerns the interpretation of phrases with source ambiguities. It is the user who generates those phrases, so he/she must know the objects and be able to link the descriptions, either linguistic or graphical, to the objects. However, the IMIG system has to identify the appropri-

ate objects, either in the domain or on the screen, as the referents of the language descriptions.

2.6 Summary

In this chapter, we review several intelligent multimodal systems as to their methods and abilities in the following aspects:

- advantages of different modalities and multimodal integration
- natural language phrases and gestures
- knowledge management
- multimodal reference processing

Each modality has advantages and disadvantages in representing certain information. Integration means different modalities can be combined in a way that the advantages of one modality can cover the disadvantages of another.

Natural language and gesture are two modalities often used in intelligent multimodal systems. Previous systems use the assumption that pointing actions happen within the time range of entering the corresponding phrases for connecting information from natural language modality with that from pointing gestures. However, this assumption is questionable in practical situations, so the IMIG system uses Hayes' method, which is based on a more reasonable assumption.

Different systems have different ways of organising their knowledge. Some systems think that entities on the screen are in the context, so they put those entities in the context model. However, we questioned the appropriateness of this arrangement in Section §2.4.6. In the IMIG system, the entities on the screen are stored in the display model, and they are added into the context model only when they are mentioned or manipulated in the interaction. In addition, the IMIG system represents the visual display attributes of screen entities in the display model and the mapping relations between screen entities and domain entities in the mapping model. These are needed in the resolution of source and referential ambiguities.

Most previous systems aim at resolving phrases with pointing actions, so their resolution processes have mechanisms for combining information from language, gesture and graphics to improve the processing. The target of the IMIG system goes beyond this to include source ambiguities. It organises the knowledge to facilitate the resolution, and has a constraint satisfaction based method to resolve source and referential ambiguities together.

Several important features of the IMIG system have been mentioned in this chapter, which distinguish the IMIG system from previous ones. Starting from Chapter 3, we will present the details of these features.

Chapter 3

Source Ambiguities and Referring Expressions

In Chapter 1, we briefly introduced the theme of this thesis, and mentioned that we were particularly interested in building a computational model capable of dealing with source ambiguity problems. However, we did not provide much detail about the nature of those problems and our proposed method to resolve them. These are the topics of this chapter. Firstly, we are going to examine source ambiguity problems in depth to reveal their nature. Then we will give some semi-formal definitions of the terms used in describing and resolving the source ambiguities. Finally, we will investigate the linguistic regularities within and between referring expressions and sentences that are related to the source aspect of the phrases. Based on these regularities, some heuristic rules can be extracted.

3.1 Source and source ambiguities

3.1.1 Source ambiguities with more examples

In Section §1.2, we mentioned several reference processing problems that were called *source ambiguities*. These problems, as reviewed in Chapter 2, cannot be systematically resolved by way of the current literature. In fact, they have not been well explored before. In this section, we would like to discuss source ambiguities in more detail through some examples.

Suppose Figure 3.1 and Figure 3.2 are screen displays for (3.1) and (3.2) respectively. The arrows and the letters in brackets, i.e., (c) in Figure 3.1 and (a) in Figure 3.2 represent the pointing actions that happen in the corresponding sentences, i.e., (c) in (3.1) and (a) in (3.2). Both of these dialogues happen in the imaginary car selection system. As previously mentioned, icons on the screen represent individual cars, and various characteristics of the icons convey attributes of the corresponding cars.

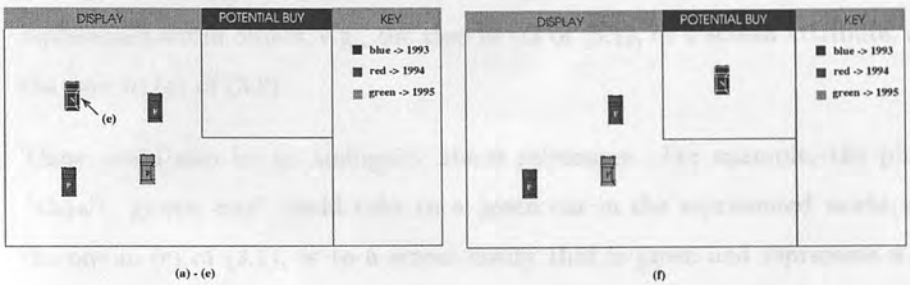


Figure 3.1: Screen displays for (3.1) of the Car Selection System

(3.1)
User: Is there a green car in your stock? (a)
System: `Yes, we have quite a few. Here, all the green cars are displayed on the screen.` (b)
User: How much is this$\nwarrow_{(c)}$ green car? (c)
System: `It is 2,400 pounds.` (d)
User: It is a reasonable price. Move it to the potential buy area. (e)
System: `The icon has been moved.` (f)

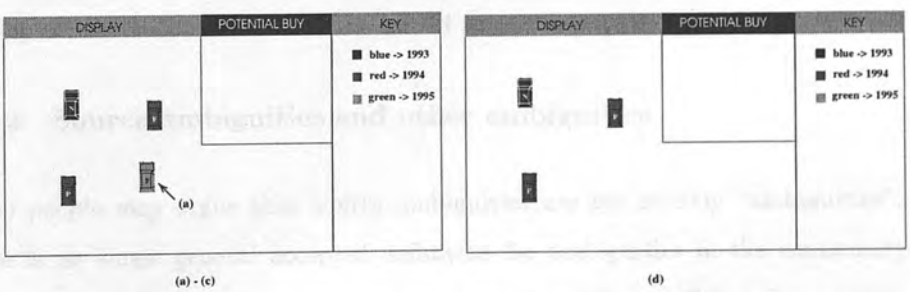


Figure 3.2: Screen displays for (3.2) of the Car Selection System

(3.2)
User: How much is this$\nwarrow_{(a)}$ green car? (a)
System: `It is 4,300 pounds.` (b)
User: It is too expensive. Remove it. (c)

System: `The green icon has been removed.` (d)

These two dialogues illustrate several interesting problems. For instance,

1. There could be an ambiguity involving the exact meaning of a word. That is, the attribute **green** in the phrase "`this↖ green car`" could be a feature of the represented world object, e.g., the case in (c) of (3.1), or a screen attribute, e.g., the case in (a) of (3.2).

2. There could also be an ambiguity about references. For example, the phrase "`this↖ green car`" could refer to a green car in the represented world, e.g., the one in (c) of (3.1), or to a screen entity that is green and represents a car, e.g., the one in (a) of (3.2).

3. There could even be an ambiguity about the meaning of the whole sentence. For example, the sentence "`move it to the potential buy area`" could either request an event on the screen or an action in the represented world in some circumstances.

The above problems, though appearing in several forms, share a common cause. That is, whether a word (e.g., an attribute of an entity), a referent (e.g., an entity referred to by a phrase) or an action applies to an entity belongs to the world model or to the display model. In other words, they are all *source ambiguity problems.*

3.1.2 Source ambiguities and other ambiguities

Some people may argue that source ambiguities are not strictly "ambiguities", but there is no single general accepted definition for ambiguities in the community. It seems reasonable to describe a sentence or phrase as ambiguous if there is more than one clearly defined meaning for it. Source ambiguities occur at the semantic processing level (cf. [Gazdar and Mellish, 1989, Section §10.1]), and, as shown in the above examples, they affect various aspects of the semantic level, including referent identification. So, we think it is reasonable to call them ambiguities.

Source ambiguities are related to referential ambiguities and lexical ambiguities. A referential ambiguity is a kind of semantic ambiguity [Russell and Norvig, 1995, p. 681]. It occurs when more than one entity can be the referent of a referring expression. Some source ambiguity can be defined as a referential ambiguity since there are more than one entity to be the referent if entities in both sources are taken into account. A lexical ambiguity occurs when a referring word has more than one meaning [Carter, 1987, p. 15]. Some source ambiguity can be viewed as a lexical ambiguity as well because some word can have more than one meaning by considering its possible sources. However, since previous research on lexical and referential ambiguities mainly concerns scenarios involving only domain entities and attributes, we distinguish a source ambiguity from these two types of ambiguity to emphasis that entities and attributes from various sources (two in our case) are involved. In some sense, we can say that source ambiguities are extensions to lexical and referential ambiguities in multi-source situations.

Our method for resolving source ambiguities reflects our view of source ambiguities. In Section 3.11, we will show that our analysis of texts with source ambiguities provides us with a set of heuristic rules. These rules are unconventional in the sense that most of them consider the source of words, and this has not been addressed in the literature. However, the ideas behind most of the rules are shared by previous research on lexical and referential ambiguities. So in some sense our work provides extra information for resolving those lexical and referential ambiguities in multi-source situations (i.e. source ambiguities). Another link is the computational model we have developed for resolving source ambiguities. It has its root in the literature of reference disambiguation (see Chapter 5). It is a model based on a constraint satisfaction method, which has been successfully applied to resolving referential ambiguities [Mellish, 1985, Haddock, 1988]. Our model extends the work of Mellish & Haddock by integrating the rules we have found for source ambiguities so that the model can handle source and referential ambiguities in multi-source situations.

However, we do not claim that our method is the only way to handle source ambiguities. People who like statistics may build a disambiguation program based on statistical methods similar to what they have proposed in word-sense disambiguation [Gale et al., 1992]. The reason that we do not use statistical methods for source

disambiguation is that we do not have a corpus to build a language model. People familiar with formal semantics may think that logical typing is a way of handling source ambiguities. Indeed, our definition of source and the way that it is used in the meaning representation language MRL_S (see Chapter 4) show some traces of the ideas of logical typing. However, apart from this, our method is different to the traditional way of using logical typing, that is, to use the syntax of the type language to resolve ambiguities [Bunt, 1985]. Our considerations are:

1. as the first attempt to handling source ambiguities and the fact that we do not have a corpus for dialogues with source ambiguities, the rules we have discovered are more appropriate to be heuristics that could be ignored sometimes rather than obligatory syntax restrictions of a language.

2. there are connections between source disambiguation and resolving referential ambiguities (see the discussion in Chapter 5). Therefore, resolving source ambiguities by typed language parser whereas resolving referential ambiguities at a different module could pose an difficult requirement to build an efficient communication channel between the two modules. In this case, our approach of resolving all the source and referential ambiguities together in the reference module is better than logical typing because the effects of resolving one ambiguity can be easily spread to the resolution of others.

No matter which method is used to resolve source ambiguities, the heuristic rules presented in this chapter give essential pieces of information.

3.1.3 Screen, domain and mixed-source phrases

The interpretation of dialogues with source ambiguities may require information from resources other than the words in the dialogues. For example, two such important information resources are pointing actions and the context of dialogues. In Chapter 1, we introduced the presentation device $\nwarrow$ to indicate the associated word of a pointing action. Here we introduce another presentation device adopted in this thesis, a subscript attached to a word. The subscript can be s or d, representing the screen or the domain. It indicates one piece of information that cannot be presented by the word

itself – the source of a word in the context of a scenario assumed in the presentation of an example dialogue.

In a referring expression, except for some function words, most words can have a source in a dialogue scenario, and the source subscripts are useful here to achieve conciseness in presenting the words and their sources in that scenario. For the convenience of discussion, we say that a phrase is a **screen phrase** in an example scenario when all the words of the phrase in the scenario have a screen subscript, e.g. "this$_s$ small$_s$ icon$_s$". Accordingly, phrases whose words all have a domain subscript are called **domain phrases**, e.g. "the expensive$_d$ car$_d$". It is possible that some words in a phrase are from the domain, whereas others are from the screen. We call such phrases **mixed-source phrases**. For instance, "the blue$_s$ car$_d$" is a mixed-source phrase.

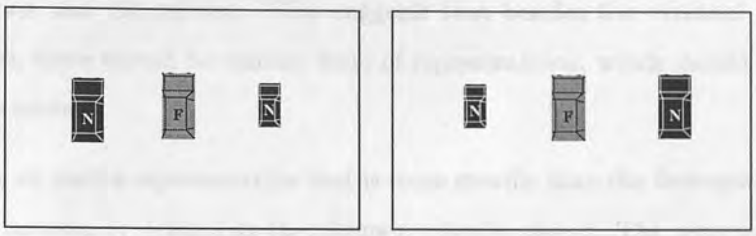
It might be odd to think about mixed-source phrases at first glance, but their usage is supported by Dale's three principles of reference [Dale, 1992, p. 117], which are based on Grice's conversational maxims [Grice, 1975]. The principles of **sensitivity, adequacy**, and **efficiency** show that the speaker would use relatively efficient ways to present adequate information about an object so that the referent is unambiguously identifiable in the context. So when a mixed-source description that can satisfy the goal is more appropriate, the speaker might use it.

3.2 The necessity of the described entity set

The *descriptive contents* of referring expressions, a term in Kronfeld's terminology [Kronfeld, 1990], are typically treated as being a function of the meanings of the words in the referring expressions. It is essentially the traditional philosophical notion of *sense*. Like pragmatic restrictions, such as constraints from world knowledge and discourse, etc. [Appelt, 1989], the descriptive contents provide the constraints that must be satisfied by the referents. The descriptive contents and the referents provide a representation structure of the referring expressions.

However, this structure has problems. It is well accepted that the same descriptive content of a phrase can correspond to different referents in different situations. An example is given in (3.3).

(3.3) **User:** Is the dark car to the right of the grey car small? (a)
System: `Yes, it is.` (b)
Action: *The user swaps the two dark cars by using mouse. The screen display then changes from (A) to (B) (see Figure 3.3)* (c)
User: Is the dark car to the right of the grey car small? (d)
System: `No, it is a large car.` (e)



(A) (B)

Figure 3.3: Screen displays for (3.3) of the Car Selection System

In the circumstances where source ambiguities could happen, the problem can be exaggerated to a kind of *reference disorientation.* That is, the descriptive content of a referring expression in a sentence may appear to describe an entity in one source, but when considered in a larger context, e.g. with other components of the same sentence, it may actually refer to the corresponding entity in the other source. For example, the phrase "`the expensive car`" can be identified as describing a domain entity that is a car which is expensive, and it refers to such an entity in the sentence "`buy the expensive car`". However, it functions differently in the sentence "`move the expensive car to the top of the screen`". In order for the operation **move**, which is a screen action, to make sense, the only entity that the phrase can possibly refer to is the screen entity (the icon) that represents the expensive car in the domain, despite the fact that the descriptive material in the phrase (e.g. "`expensive`") applies only to the domain entity.

Another problem we found reveals the disorientation problem even more clearly. Consider the example "`move the expensive car to the top of the screen, and place an order for it`". The pronoun "`it`" seems to have "`the expensive car`" as its antecedent according to linguistic intuitions, otherwise it is difficult to interpret the pronoun. However, if we examine the referents of the two phrases, we encounter a prob-

lem. The referent of the pronoun "`it`" is an expensive car in the domain, while that of the phrase "`the expensive car`" is the screen icon corresponding to that expensive domain car.

From the above examples, we know that the complication in a source ambiguity situation comes from the lack of a direct descriptive relation between the components of the phrase and the referent. This suggests that besides the eventual referent of each phrase, there should be another form of representation, which should not be the descriptive content.

Informally, we need a representation that is more specific than the descriptive content, but is not necessarily defined as the ultimate referent object. The representation we selected is a set of item(s) described by the descriptive content (sense) of a phrase in the current situation. This allows both the described item and its corresponding entity (i.e. the entity connected with the described item through the mapping relation) to be considered in the process of finding the ultimate referent object, thus resolving the above problems. This set is called the **described entity set** of the referring expression, and the items in the set are called the **described entities**.

We also define each referring expression as having an **intended referent**, the item which is ultimately denoted within the context, for example, as an argument to a verb or a preposition. The intended referent is essentially what is called the *referent* in the literature.

The difference between the descriptive content of a phrase and the described entity set of the phrase is that the descriptive content is still an abstract concept, while the described entity set is a concrete set whose items are entities either from the domain or from the screen. Considering the entities available in the context, the restrictions about the sources, and the meaning of words in a phrase, the number of the described entities should usually be on a countable scale since we only consider a singular phrase (see Section §1.3.2).

In an earlier paper [He et al., 1997], we called the described entity set the *described referents*. We modified the name in another paper [He et al., 1998] and in this thesis in order to avoid irrelevant philosophical disputes about what can be called referents.

The described entity set is merely a convenient formal construct in our theory.

3.3 Semi-formal definitions

In Section §3.2, we gave informal definitions of the described entity set and the intended referent of a referring expression. For the clarity of our model, more elaborated discussion is needed for both the definitions and their relations. In the following, we give semi-formal definitions to elaborate the relation between described entity sets and intended referents.

Although all these definitions would have to be generalised to cover plural noun phrases in the future, the concepts are made clearer by considering only singular noun phrases at the moment.

Definition 1 *A* **described entity set** R *is a set with one of the following forms:*

1. *a singleton set containing either a domain entity or a screen entity;*
2. *a two-element set containing a domain entity and a screen entity related by the representation mapping.*

As mentioned, the described entity set has to be computed from the descriptive content. The problem now is how the described entity set is obtained from the descriptive content.

Halliday mentioned, in his book about functional grammar, the logical structure of nominal groups, to which referring expressions belong [Halliday, 1994, p. 191]. There is usually a *head* in a noun phrase, and there could be several *pre-modifiers* before the head and several *post-modifiers* after it. Both pre-modifiers and post-modifiers are called *modifiers*. For example, the logical structure of the phrase "`these two small blue Nissan cars to the right of the corner`" is illustrated in Table 3.1, where the head is "`cars`", the pre-modifiers are "`these two small blue Nissan`", and the *post-modifier* is "`to the right of the corner`". A common logical relation between a head and the modifiers is *modification*. That is, each modifier tries to identify an

attribute of the referent denoted by the head so that the *subcategorization* relation "*A is a subset of X*" is satisfied, where "*A*" is the referent(s) of the phrase, and "*X*" is the object class identified by the head.

these	two	small blue	Nissan	cars	to the right of the corner
		Pre-modifier		Head	Post-modifier

Table 3.1: The logical structure of a referring expression [Halliday, 1994, p. 191]

The relations between modifiers can be complicated (for example, *submodification*, in which a modifier modifies another modifier rather than the head [Halliday, 1994]). In this thesis, however, we assume that modifiers in a noun phrase are semantically independent from each other. That is, the descriptive content of a noun phrase is a conjunction of the independent meanings of each word. Therefore, the relation between the descriptive content of a noun phrase and its described entity set can be defined as the following:

Definition 2 *The descriptive content* DC *of a noun phrase* **describes** *a described entity set* DE *iff*

1. *every predicate in* DC *is true of an element in* DE, *with source taken into account (i.e. domain predicates can only be true for domain entities, and screen predicates are only be true for screen entities); and*
2. *for every element in* DE, *there is a predicate in* DC *which is true for the element with source taken into account.*

For example, the phrase "`the small`$_d$ `expensive`$_d$ `car`$_d$" would "describe" the described entity set {`car1`} if **car1** is an entity which is classified as a **car** and has the properties **small** and **expensive** asserted in the world model. Another more complex example is that the described entity set of the phrase "`the red`$_s$ `car`$_d$" is {`icon3, car2`} if **car2** is an domain entity which is classified as a **car**, **icon3** is an entity on the screen with **red** screen colour attribute, and, in addition, **car2** and **icon3** are the corresponding entities of each other. Notice that this relationship between phrases

and entities is relatively context-free linguistically, in that it does not take into account the surrounding sentence structures (if any). However it takes into account the environment by checking and accepting a domain entity into the described entity set.

Definition 3 *A* **described entity set assignment** *is a mapping which allocates to each noun phrase* N *a described entity set* DE *such that the semantic form of* N *describes* DE.

Informally, the described entity set of an NP contains either an entity that is completely described by the content of a single-source NP, e.g., **car1** by the phrase "$\texttt{the small}_d$ $\texttt{expensive}_d$ $\texttt{car}_d$", or two corresponding entities each of which is described by a mixed-source phrase, e.g., **car2** and **icon3** by "$\texttt{the red}_s$ $\texttt{car}_d$". Definition 3 is intended to capture the notion of a "reading" or an "interpretation", so that we can discuss the phenomena we are interested in without complications from other forms of ambiguity. These other factors may give rise to alternative described entity set assignments.

To capture the effect that a phrase can refer to the entity corresponding to (one of) the entities that it actually describes, we need two more definitions.

Definition 4 *Given a described entity set* DE, *the* **potential referents** *of* DE *is the set:* $\text{DE} \cup \{y \mid$ *there is an* x *in* DE *which is related to* y *by the representation mapping*$\}$.

Definition 5 *Given a described entity set assignment* DEA, *an* **intended referent assignment** *based on* DEA *is a mapping* IRA *which allocates to each noun phrase* N *an element of the potential referents of* DEA(N).

In Definition 5, IRA(N) is a formalisation of the intended referent of a phrase N.

Definitions 1 – 5 reveal a chain starting from the descriptive content of a phrase toward the intended referent of the phrase. Located in the middle of the chain, the described entity set has much closer relation to the intended referent, and the relationship is still flexible enough to allow non-one-to-one relations. That is, the intended referents may either be the same as the described entities (as in "$\texttt{Buy}_d$ $\texttt{the expensive}_d$ $\texttt{car}_d$") or correspond, via the representation mapping, to the described entities (as in "$\texttt{Move}_s$ $\texttt{the expensive}_d$ $\texttt{car}_d$ $\texttt{to the top}_s$ $\texttt{of the screen}_s$"). This is exactly what we want.

The definitions of described entity set and the intended referent also indicates that different phrases whose described entity sets are different could have the same potential referent set, which means that their intended referents could be the same. For example, the phrases "the $\texttt{expensive}_d$ $\texttt{car}_d$" and "the $\texttt{blue}_s$ $\texttt{icon}_s$" can have different described entity sets $\{\texttt{car}_1\}$ and $\{\texttt{icon}_1\}$, respectively. But under the condition that $\mathbf{car}_1$ and $\mathbf{icon}_1$ are corresponding entities, their potential referent set is the same, i.e., $\{\texttt{car}_1, \texttt{icon}_1\}$. This means that their intended referents could also be the same, i.e., either $\mathbf{car}_1$ or $\mathbf{icon}_1$.

For phrases where there is a single source for their words, there is no doubt about the source of the described entity sets. For mixed-source phrases, there could be a pair of corresponding entities from different sources in the described entity set. For this reason, we do not talk about the source of the described entity set, although the described entities in the set have their sources, like the intended referents.

3.4 Referring expressions in source ambiguities

Source ambiguities are related to entities, the attributes of entities and the actions applied to entities. In terms of linguistic categories, this means that most source ambiguities are related to nouns, adjectives, prepositions, and verbs. Among all sorts of *word complexes*, a term means "*a combination of words built up on the basis of a particular logical relation*" [Halliday, 1994], the nominal group is the one that covers most of these categories. This is why we concentrated on analysing nominal groups.

Many types of referring expressions could appear in the interaction. However, as mentioned in Chapter 1, the referring expressions in our consideration are those referring to singular concrete entities that exist in the system's models; the pre-modifiers contain epithet adjectives and/or a noun classifier, and the only type of post-modifiers allowed are prepositional phrases. In addition, we assume that there is no *one*-anaphora in a input sentence.

There could be two distinct but not necessarily mutual exclusive ways of approaching the resolving of referring expressions. The first approach examines the speaker's intentions, goals, and other high level features which are relevant to the generation

of referring expressions, e.g., [Kronfeld, 1990]. The other approach starts from a relatively low level. It analyses the linguistic regularities among words or other functional elements of referring expressions so that some heuristic rules for resolution can be summarised, e.g., [Hobbs, 1978].

We adopted the second approach, because our intention was to have a clearer understanding of the nature of source ambiguities through some low-level analysis of referring expressions with such ambiguities. This could also help a further exploration from the communicative intention aspect.

3.4.1 Types of referring expressions

For a clear discussion of referring expressions and source ambiguities, we enumerate the types of referring expressions under consideration in the next section. There could be several ways to classify noun phrases that fall into our consideration. Each way is useful in some situations.

Classifying by source

According to the sources of its components, a noun phrase can be classified as a screen phrase, a domain phrase, a mixed-source phrase, or an unknown phrase. The first three types were mentioned in Section §3.1, and the criterion of division is the sources of the words in the phrase. Before the source ambiguities are resolved, some words in the phrase may not have a clear source, so we call the phrase an **unknown phrase**. After the resolution, an unknown phrase could become a screen, domain or mixed-source phrase. Personal pronouns such as "`it`" are defined as unknown phrases, because information about the source of "`it`" is not available before its antecedent is determined. We define the source of "`it`" as being the same as that of its antecedent.

This classification is useful when the sources of the described entities of a referring expression are in consideration, since as indicated in the definitions in Section §3.3, there are links between the sources of the words in a phrase and the sources of the described entities of the phrase.

Classifying by referent origin

This classification appears superficially to be based on which kind of determiner is used in a referring expression, but it is actually relevant to the referent origins. This is because one of the major functions of the determiners is to tell where the referents are [Halliday, 1994]. The referring expressions in our consideration are classified into four types.

- **Definite descriptions**, in this thesis, are noun phrases using the definite article *the*[1]. They generally have two groups, **anaphoric definite descriptions** and **definite descriptions with novel referents** [Vieira and Poesio, 1997]. A novel definite description introduces a new object into the dialogue, which does not have an antecedent in a previous context. In contrast, the function of an anaphoric definite description is not to bring a new object into dialogue context, but to "*point back*" to some people, place, object, time or event in previous dialogue context. Among the various usages of definite descriptions with novel referents (see [Vieira, 1997]), we are particularly interested in *visible situation use* [Hawkins, 1978], which occurs when the referent is visible to both the speaker and the listener. This usage is common in a multimodal interface, because the graphics displayed on the screen can be seen by the user and, in a metaphoric sense, by the system. Such an example can be found in (3.4), where the intended referent of "`the green car`" is provided through the icon on the screen rather than by the previous linguistic context.

(3.4) **User:** How much is the green car? (a)
System: `14,200 pounds.` (b)

- **Deictic phrases**, in this thesis, are referring expressions headed by demonstrative words like "`this`" and "`that`". Considering whether or not there are pointing actions accompanying them, they are classified into two groups: *unimodal deictic phrases* and *multimodal deictic phrases.*

  A **unimodal deictic phrase** is a deictic phrase without a pointing action. Sim-

[1] This definition follows [Vieira, 1997] and [Russell, 1905].

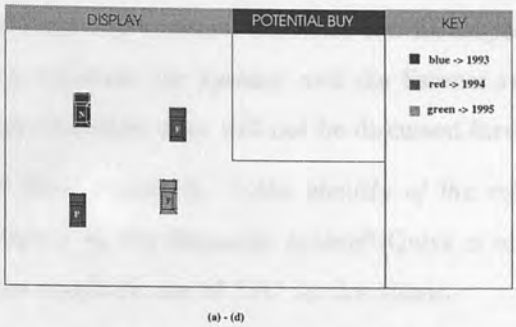


Figure 3.4: Screen displays for (3.4) of the Car Selection System

ilar to an anaphoric definite noun phrase, it is used to point back to an entity mentioned in the previous dialogue, especially when the determiner is "`that`". For example

(3.5) **User:** Move this↖ car here↖ . (a)
System: `Yes, your command succeeds.` (b)
User: Is that car small? (c)
System: `Yes, it is.` (d)

A **multimodel deictic phrase** has a pointing action besides the linguistic form. It is similar to what is called a *deictic phrase* by the intelligent multimodal community. An example is the phrase "`this`↖ `car`" in sentence (a) of (3.5).

- **Quantified phrases**, in this thesis, are noun phrases headed by quantifier words like "`a`", "`every`" and "`which`". Here we adopt Alshawi's idea in the CLE project [Alshawi, 1992] to treat interrogative words as common quantifiers.

  We treat this type of phrase differently from the above two types, and do not try to enumerate entities for the referents of a quantified phrase[2]. This is because either there is not enough information to find the referents, as for indefinite noun phrases like "`a blue car`", "`some small icons`", "`which car`" and so on; or there is no need to do so, for phrases like "`every blue car`".

- **Personal pronouns** considered in this thesis are "`I`", "`you`" and "`it`". This is due to the fact that only singular phrases are allowed and no other agent takes

[2] It could be argued that quantified phrases are not referring expressions.

part in the two imaginary systems. The first two have specific meanings in our dialogues, which represent the speaker and the listener respectively. As they always have clear meanings, they will not be discussed further.

Because "*much more frequently, ...,the identity of the referents of 3rd person pronouns is supplied by the linguistic context*"[Quirk et al., 1985, p. 347], we only consider the anaphoric use of "`it`" in this thesis.

As it is difficult to define the described entity set for "`it`" based solely on the phrase itself, we define its described entity set as being the same as that of its antecedent. For instance, we have a sentence "`move the expensive car to the top of the screen and place an order for it.`" where "`it`" and "`the expensive car`" are related as anaphor and antecedent, they are defined to have the same described entity set. However, their intended referents could be different, which is the case in the example.

3.5 Restrictions within referring expressions

In the remaining sections of this chapter, we will present several linguistic regularities for the source restrictions between words. These regularities can be abstracted into heuristic rules, which are used in the resolution of source ambiguities. We have no intention to claim that they are the only linguistic regularities occurring between referring expressions, either with respect to the source aspect or beyond. They simply represent what we have found during our research.

In the discussion about the concept *describe*, the logical structure of a referring expression proposed by Halliday in his functional grammar was mentioned. In the following discussion, another structure for referring expressions, the experiential structure also proposed by Halliday [Halliday, 1994], will be used.

Slightly away from the discussion track, we want to emphasis the reason that we use the functional grammar. We are interested in how the sequences or structures of noun phrases (referring expressions) contribute to the meaning of the phrases. Functional grammar is designed for this purpose, as stated by Halliday: "*In fact the meanings are encoded in 'wordings': grammatical sequences, ... The relation between the meaning*

and the wording is not, however, an arbitrary one; the form of the grammar relates naturally to the meanings that are being encoded. A functional grammar is designed to bring this out; it is a study of wording, but one that interprets the wording by reference to what it means." [Halliday, 1994, p. xvii].

Table 3.2 is the experiential structure of an example referring expression. "deictic", "numerative", "epithet", "classifier", "thing", and "qualifier" are the functional labels for the components in the experiential structure. Some of them are self-explanatory, and the others will be discussed in detail in the rest of this section. However, "numerative" will not be discussed further, because we only consider singular referring expressions.

Deictic	Numerative	Epithet	Classifier	Thing	Qualifier
`these`	`two`	`small blue`	`nissan`	`cars`	`at the right of the corner`

Table 3.2: The experiential structure of referring expressions [Halliday, 1994, p. 191]

3.5.1 The deictic component

This use of *deictic* shall be distinguished from its use in deictic phrases introduced in Section §3.4.1. The former is a functional label of a component in the experiential structure of a referring expression, while the latter is a kind of noun phrase starting with a demonstrative word. To distinguish between them, we refer to the former as *the deictic component.*

The deictic component we considered is associated with three kinds of words:

- the definite article "`the`" when the phrase is a definite noun phrase;
- the demonstrative words "`this`" and "`that`" when the phrase is a deictic phrase; and
- quantifier words like "`a`", "`which`" and "`what`", when the phrase is a quantified phrase.

According to the functional grammar, the main function of the deictic component is to identify a subset of entities described by the other parts of the phrase, and also to give

a clue as to which subset it is. For example, the meaning of "the" is that "*the subset in question is identifiable; but this will not tell you how to identify it — the information is somewhere around, where you can recover it*", while the meaning of "this" is "*you know which object: — the one near me*" [Halliday, 1994, p. 181].

Despite the differences among the three types of words, it could be argued that the deictic component generally does not have a strong favourite for a particular source. The sources of the entities in the described entity set of a phrase depend on the remaining parts of the phrase. For example, one of the main uses of the definite article "the" is in anaphoric phrases, which means that the objects in the described entity set are mentioned in the previous context. The referents can be either domain entities or screen entities.

However, exceptions do occur. When the deictic component is accompanied by a pointing action, which directly operates on a particular screen entity, it is reasonable to assume that the screen entity is "described" by the phrase and the action, and thus should be included in the described entity set. It does not matter whether this entity is mentioned in a previous dialogue or not.

Although, phrases in multimodal references can have many forms, such as "the↖ car", "this↖ icon", and so on, to simplify the resolution process, we assume that the deictic element of a multimodal reference phrase can only use demonstrative words.

The above discussion can be summarised by heuristic RULE 3.1

> RULE 3.1: The deictic element of a phrase is a demonstrative word with a pointing action $\Longrightarrow$ the screen entity that is pointed to must be in the described entity set.

3.5.2 The thing component

The thing component *"is the semantic core of the nominal group"* [Halliday, 1994, p. 189]. It could be a personal pronoun, a proper name or a common noun.

Personal pronouns are seldom used with other functional components of referring expressions, so they are not very helpful for identifying the source restrictions within

phrases. Therefore, they are not considered in finding regularities about the source aspects.

Proper names are not included either for the following reasons. Firstly, they are seldom used in our imaginary systems. The settings of the two imaginary systems make it difficult to refer to the domain entities by proper names.

Secondly, if the user knows the internal names of entities, proper names would not cause any new source ambiguity in the interaction. As Russell claimed [Russell, 1953], proper names designate objects directly rather than through the use of the meanings of descriptions. In addition, unlike other descriptions that change their denotations in various occasions, a proper name always designates the same individual in all the possible worlds as long as this designation relation is set at the beginning. Consequently, making the reasonable assumption that each entity in the systems is individual and different, we could claim that proper names do not cause any source ambiguity.

Thirdly, at least in our consideration, proper names are also seldom used with other parts of functional components of a nominal group, so they are not helpful in identifying the source restrictions within phrases.

Common nouns are our major concern. A common noun refers to a set of individuals. Typically, it is used with words from several other functional components to narrow down the possibilities for described entities and the intended referent. A number of interesting issues about source ambiguities arise from noun phrases with a common noun as the thing component. We will call this a **head noun** for the rest of this thesis.

In Section §1.3.2 and §3.1, we mentioned the representational mapping relations between screen entities and domain entities. These relations are asymmetric: for example, we can say that "`a square represents a car`", but we cannot say that "`a car represents a square`". This asymmetry restricts the use of head nouns in the interaction. Intuitively, a screen entity such as a **square** can be referred to by a head noun naming its screen category, i.e., "$\texttt{square}_s$", or by a word naming the category of the domain entity it represents, i.e., "$\texttt{car}_d$". However, a domain entity such as a **car** cannot be referred to by a head noun naming the corresponding screen category, that is, it is very odd to refer to a **car** as a "$\texttt{square}_s$". Hence, the heuristic rule about the

intended referent of a phrase and the head noun of the phrase is:

> RULE 3.2: The head noun of a phrase unambiguously names a screen category $\Longrightarrow$ the intended referent of the phrase must be a screen entity.

If the head noun of a phrase is a domain word, there is no restriction about the source of the intended referent. According to the fact that there are only two sources, i.e. the screen and the domain, a statement deduced from RULE 3.2 is that the head noun of a phrase must be from the domain if the intended referent is a domain entity.

RULE 3.2 is a necessary condition for a meaningful sentence. If sentences violate the rule, they are unacceptable in our domains. For example,

(3.6) * Buy$_d$ the icon$_s$.
* Is the square$_s$ expensive$_d$?

3.5.3 The epithet, classifier and qualifier components

The epithet, classifier and qualifier components are the functional elements of noun phrases. According to the logical structure of a noun phrase (comparing Table 3.1 and Table 3.2), they are *modifiers* to the head noun of the phrase.

Therefore, the epithet, classifier and qualifier components provide extra information to distinguish an entity in the class specified by the head noun from the other objects in the same class. Hence, the attributes provided by the modifiers should belong to the class of entities specified by the head noun. RULE 3.2 indicates that a screen head noun can denote only a class of screen entities, so this restricts the modifiers of such head noun to denote only screen attributes. As a result, those modifiers must be screen words. For example, while phrases like "a red$_s$ icon$_s$" and "a small$_s$ circle$_s$" are meaningful expressions, phrases like "*the expensive$_d$ icon$_s$" and "*every large$_d$ square$_s$" are unacceptable. In contrast, if the head noun is from the domain, there is no strong restriction on modifiers. So, phrases like "the expensive$_d$ car$_d$ in$_s$ the corner$_s$", "the red$_s$ car$_d$", "a small$_d$ saloon$_d$" are all acceptable.

However, there are exceptions. A domain word acting as the classifier component (especially if it is a noun) can sometimes be so closely bonded to the head noun

denoting a fairly general screen class that the whole phrase is still meaningful in that combination. For example, "`the Nissan`$_d$ `icon`$_s$" is one. The reason could be that the classifier component is not functioning as a modifier in this case, but helping to form a compound head noun. We have not found any strong regulation about classifier modifiers, so we will only talk about non-classifier modifiers in the following heuristic rules.

RULE 3.3 is a heuristic rule about the head noun and the modifiers summarised from the above discussions.

> RULE 3.3: The head noun of a phrase unambiguously names a screen category $\Longrightarrow$ the described entities described by the non-classifier modifiers of the phrase must be screen entities.

A deduction of RULE 3.3 is that the described entity set of a phrase without classifier modifiers would contain only screen entities if the head noun satisfies RULE 3.3.

RULE 3.2 and 3.3 both talk about the situations when the head noun unambiguously names a screen category, so we can combine them as the **screen head noun rule**:

> RULE 3.4:
>
> **[Screen head noun rule]** The head noun of a phrase unambiguously names a screen category $\Longrightarrow$ the intended referent of the phrase must be a screen entity and the described entities described by the non-classifier modifiers of the phrase must be screen entities as well.

3.6 Intra-sentential source influence between referring expressions

Before we go on to talk about the intra-sentential source influence, we need to explain what is our view of organising concepts in hierarchical structure.

3.6.1 The hierarchy

A hierarchical structure is used in many knowledge base systems to reflect relations among concepts/semantic categories (e.g. [Bateman et al., 1995, Beierle et al., 1990]). It organises concepts into a tree structure. From the root of the hierarchy, concepts go from abstract ones down through less abstract nodes to more concrete leaves. The concepts in our hierarchy are divided into the following three classes:

Class 1: concepts for entities and attributes/relations/operations that only appear in the world model, such as `cars, prices`.

Class 2: concepts for entities and attributes/relations/operations that only appear in the display model, such as `icons, delete`.

Class 3: concepts for entities and attributes/relations/operations that appear in both models, such as `colours, size`.

The benefit of this arrangement is that the hierarchical information can help in identifying the source for entities and features. If an entity or an attribute/relation/operation is a sub-category of a concept that represents those only appearing in the world model or in the display model, its source is the domain or the screen respectively. Otherwise, the source is unclear and we need more information to identify it. Further information about the hierarchy can be found in Section §6.7.2.

3.6.2 The heuristics about intra-sentential source influence

Besides the regularities mentioned in Section §3.5, we also observed restrictions about the source aspect between different phrases in a sentence, although the effects of the latter are probably weaker by comparing with that of the former.

In the past decades, many psycholinguistic observations of human generated referring expressions were performed. Levelt gave a review of the achievements [Levelt, 1989], in which one point is of particular interested to us, that is, "*the speaker apparently contrasted the new referent object with the previous one*". This means that comparisons between features are used while referring. Common sense tells us that features used

during comparison are different but comparable. As it is much easier to let the comparison be understood if the attributes used in comparison are from the same source rather than from different sources, we can propose the following weak heuristic rules:

RULE 3.5:

[Same type rule] two words in the same sentence share the same semantic type $\Longrightarrow$ they probably have the same source.

Intuitively, hierarchies like the one introduced in the last section is a good place to consult for relations between semantic categories. Various criteria can be used to judge if two semantic categories are of the same type. A straightforward method is to see if there is a concept directly subsuming both in the hierarchy. We adopt this method for our implementation of the IMIG system (more details in Chapter 6).

RULE 3.6: **[Same position rule]** two words are at the same position of two phrases, and the two phrases have the same structure $\Longrightarrow$ the two words probably have the same source.

Although the above two rules sound similar, they are actually aimed at different situations. For instance, two predicates "`red`" and "`blue`" in the sentence "`is the upholstery in the red car blue?`" are probably from the same source because their relation satisfies the *same type* rule. An example in which the relation satisfies the *same position* rule is the sentence "`move the small car to the right of the expensive car.`" Here, the predicates "`small`" and "`expensive`" occur in the phrases with the same structure, which suggests that they probably share a common source. In some situations, both rules can be satisfied and a conflict may happen. Section §3.11 will say more about this.

3.7 Inter-sentential source influence

In a dialogue containing just a few sentences between the two participants, i.e., the dialogue is short, we observe another source consistency. The same words in different sentences, generated by the same user, are probably from the same source.

In many cases, the two appearances of the same word belong to two phrases that are coreferential or quasi-coreferential (see Section §3.8 for the definition of quasi-coreference). This implies that these two appearances of the word are probably from the same source (see (3.9) for an example).

There are also cases that the appearances of a word in non-related phrases from the same user in different sentences are from the same source too. For example, (3.7) is a dialogue happening in the room arrangement domain. In this short dialogue, the user utters the word "`above`" twice (once in (c) and the other in (e)). By considering the fact that the second "`above`" is mentioned soon after utterance of the first "`above`", we think the sources of the two are probably the same. Again, this can be made a little generalised and summarised as a heuristic rule.

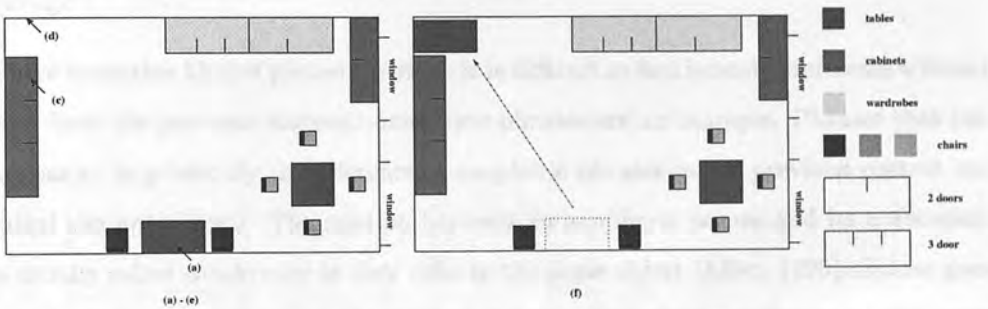


Figure 3.5: Screen displays for (3.7) of the Room Arrangement domain

(3.7)
User: What is the size of this $\nwarrow_{(a)}$ table? (a)
System: `53cm x 79cm.` (b)
User: What is the distance between this $\nwarrow_{(c)}$ cabinet and the wall above it? (c)
System: `You mean this` $\nwarrow_{(d)}$ `wall? The distance is 60cm.` (d)
User: Move the table to the place above the cabinet. (e)
System: `The table has been moved.` (f)

RULE 3.7:

a word appears more than once in adjacent sentences and all of the appearances are generated by the same user $\Longrightarrow$ all these instances of the word probably have the same source.

Of course, this is a very weak heuristic and the way that it is expressed in terms of the

nearness of two sentences makes it vague as well. However, having it has the advantage of making a selection when there is no evidence against it. We will refer to this again in Chapter 6 when we talk about the implementation.

3.8 Coreference and quasi-coreference

In a discourse or dialogue environment, the interpretation of some noun phrases is independent of the previous context. The phrases either introduce their referents directly, say by using a proper name "`Edinburgh`", or have enough descriptive contents to identify the referents directly so that there is no need for extra information from the previous context. An example is "`the most expensive car in John Smith's garage`".

There is another kind of phrase for which it is difficult to find intended referents without help from the previous context. Anaphoric phrases are an example. Phrases that can appear to help identify the referents of anaphoric phrases in the previous context are called the *antecedents*. The relation between an anaphoric phrase and its antecedent is usually called *coreference* as they refer to the same object [Allen, 1995]. Sidner goes a little further to distinguish the relation between two phrases referring to the same cognitive entity as *cospecification* [Sidner, 1987]. However, in this thesis, we will not make such distinction.

3.8.1 Definitions

In research about natural language dialogues, the coreferential relation is an important linguistic resource for resolving the referents of anaphoric phrases [Grosz, 1977, Sidner, 1987]. It is also true in studies of multimodal dialogues.

As the referent of a referring expression is called the intended referent of the phrase in our research, our definition of the coreferential relation is:

Definition 6 *Noun Phrases N1 and N2 are* **coreferential** *iff they have the same intended referents.*

However, the coreferential relation alone is not enough for our situation. Extra relations need to be defined to cope with situations where entities can come from both the screen and the domain, and some of the entities have mapping relations with entities in the other source. For example, consider the dialogue:

(3.8) **User:** What is the most expensive$_d$ car$_d$? (a)
System: `The ACMD-4000`$_d$. (b)
User: Delete$_s$ that icon$_s$. (c)

Here, "`the ACMD-4000`$_d$" and "`that icon`$_s$" could be argued to be related coreferentially or through some similar relation in the sense that the latter takes its meaning anaphorically by using the former as an antecedent. However, the two phrases have different intended referents – the first denotes a domain entity and the second a screen entity. In order to capture the specific relations between phrases that cannot be classified as coreferential but are very close to it, we introduce the concept of *quasi-coreference.*

Definition 7 *Noun Phrases N1 and N2 are* **quasi-coreferential** *iff their intended referents correspond under the representation mapping.*

Quasi-coreferential relations happen in our IMIG system from time to time. Another example can be found in the following dialogue, where the relation between "`the blue car`" and "`that blue car`" in the user's input are quasi-coreference.

(3.9) **User:** Which car is more expensive, the blue car or the red car? (a)
System: `The blue car.` (b)
User: Delete that blue car. (c)

Quasi-coreference emphasises on the situation that the intended referents of two phrases are two entities with different sources but connected by a mapping relation. This is the difference between it and coreference. However, just like what happens in coreference related to anaphora, the interpretation of the anaphoric phrase usually depends on information from its antecedent by quasi-coreference. Viewing quasi-coreference as an extension of coreference, the explanation of quasi-conference cannot be achieved by analysing some nouns as ambiguous because that of coreference cannot.

The "*strict-identity*" and "*sloppy-identity*" puzzle mentioned in natural language semantics [Dowty et al., 1981, Partee and Hendriks, 1997] may resemble quasi-coreference in some aspect, but they are different in that those two identities appear in the interpretation of ellipses and would be resolved once the ellipses are resolved. Quasi-coreference, however, can appear in wider situations. For example, the quasi-coreferences in (3.8) and (3.9) are not assocated with ellipses. In addition, the resolution of quasi-coreference needs extra information from the mapping relations.

Nevertheless, despite the above discussion, in principle the logical typing approach to source ambiguities might be capable to interpret quasi-coreference, but we do not know how to do it.

3.8.2 Heuristics

Similar to the definition of coreference in the literature, our definitions contain no details of computation. In order to identify whether or not two phrases are (quasi-) coreferential, we have to develop computational rules. Various heuristics have been given in the literature for finding the antecedents of definite descriptions. For example (adopted from [Vieira, 1997]),

- The head noun of a definite description matches the head noun of its antecedent;
- The pre-modifiers of a definite description are a subset of the pre-modifiers of its antecedent;
- A non-premodified antecedent can match any same head definite description.

In Vieira's discussion about the matching between the head noun of a definite description and that of its antecedent, the meaning of *matching* is that either the two heads are the same, or one is a generalisation/specification of the other in some taxonomy.

In our situation, the above heuristics still work, but need to be extended to handle quasi-coreference.

(3.10) **User:** Move this↖ green icon to the potential buy area. (a)
System: `The green icon has been moved.` (b)

User: Print more information about that car. (b)

As shown in (3.8), (3.9) and (3.10), the head noun of an anaphoric phrase and that of its antecedent, no matter whether coreferential or quasi-coreferential, can not only be related along a taxonomy, i.e., a generalisation, the same, or a specification, but also be from two different sources and connected by a mapping relation, e.g., between "$\texttt{car}_d$" and "$\texttt{icon}_s$", and there is a mapping relation that a domain car is represented by a screen icon.

We expect a similar thing to happen between pre-modifiers of the phrases as well. For example, the anaphoric phrase "$\texttt{the green}_s\ \texttt{car}_d$" sometimes could have an antecedent like "$\texttt{this expensive}_d\ \texttt{car}_d$", if there is a mapping relation between the screen property "$\texttt{green}_s$" and the domain property "$\texttt{expensive}_d$".

In summary, no matter whether the relation is coreferential or quasi-coreferential, the heuristics are:

RULE 3.8:

For an anaphoric phrase and its antecedent:

- The head nouns of the two phrases are the same, or one is a generalisation/specification of the other, or they have a mapping relation between them.
- Suppose set A contains attributes represented by pre-modifiers of the anaphoric phrase, set B contains the attributes represented by pre-modifiers of the antecedent, set C contains the attributes that have mapping relations with those in set A, and set D contains attributes that have mapping relations with those in set B, then the union of A and C is a subset of the union of B and D.

These heuristics can be justified by the example phrases in (3.8), (3.9) and (3.10).

The above heuristics assume that the anaphoric phrase is a definite phrase, so they are not helpful when the phrase is actually a personal pronoun, like "it". Although

the described entity set of a personal pronoun is defined to be the same as that of its antecedent in our approach, the relation between "it" and its antecedent can be coreferential or quasi-coreferential. (3.11) gives an example of a quasi-coreferential relation. Since "it" itself provides little information, its resolution depends on its antecedent. Being different to a definite description, "it" has no restriction about its source aspect, so we use the existing methods in the literature to find antecedents [Ersan and Akman, 1994, Sidner, 1987]. The algorithm for resolving pronouns will be presented in Chapter 6, as a part of the implementation.

(3.11) **User:** Which car is more expensive, the blue car or the red car? (a)
System: The blue car. (b)
User: Delete it. (c)

3.9 Heuristics on salience and position

In this section, we are going to talk about two heuristics that indicate priorities in selecting intended referents. Like other heuristics mentioned in Section §3.6 and onwards, they are formalised as preferences in the resolution model handling source ambiguities (see Chapter 5).

Salience is a property of discourse entities. Many previous researches (such as [Walker, 1997]) have linked the salience of a dialogue entity with the likelihood of that entity being the referent. That is, the more salient a dialogue entity is, the more likely it is to be the referent. This relation is usually used as a heuristic due to the complexity of human dialogues.

Salience is also a useful information in source disambiguation because it can help us to find the right intended referent, which then could provide extra clues to source disambiguation. Although there is no consistent definition of salience in the literature, the consensus among researchers is that salience is determined by a diversity of factors with varying importance [Huls et al., 1995]. Our implementation of calculating salience considers the factors of recency of mention, syntactic and semantic parallelism, the role in sentences and forgetness (see Section §6.7.7). However to keep our discussion in this section at an appropriate abstract level, we present the following heuristic rule without

giving any detail of how the calculation of salience is implemented:

RULE 3.9:

When selecting from several discourse entities that are available to be the intended referent of a referring expressions, the one that has highest salience is preferred.

The second heuristics is also about selecting an entity to be the intended referent of a referring expression. It talks about the spatial relation between an entity and the pointed position. In the intelligent multimodal interface literature, people usually assume that the nearer an entity is to the pointed position, the more likely it is to be the entity that the user wants to pick up [Neal and Shapiro, 1991]. Therefore, this rule is written as follows:

RULE 3.10:

When selecting from several screen entities that are available to be the intended referent of a referring expressions, the one that is nearest to the pointing position is preferred.

3.10 More specific heuristics

The heuristics mentioned in Section §3.5 to 3.9 are domain independent and they should be applicable to both a non-spatial domain like the Car Selection domain and a spatial domain like the Room Arrangement domain. In addition, there are rules of thumb that are restricted to a specific domain.

When exploring a domain in some details, many domain specific heuristics can be found. For example, in the Car Selection domain, if the engine of an entity is mentioned, the entity must be a car in the domain rather than an icon on the screen. We do not want to go into such details because of their lack of generality, but instead introduce two heuristics that are valid either in most spatial domains or in most non-spatial domains.

A non-spatial domain is an application domain which does not have spatial relations among its domain entities or does not consider its domain spatial relations in the interaction. An obvious heuristic for such a domain is that spatial relations probably have the screen source.

In a spatial domain, because the spatial relations from both the domain and the screen can be involved in the interaction, no such obvious heuristic is available. In such a domain, the domain spatial relations can only be perceived by the user when they are depicted by the screen spatial relations. Since the user can see screen spatial relations directly, but can only perceive the domain spatial relations through a cognitive transformation process from the screen spatial relations to the domain relations, it could be a preferred that a spatial relation mentioned in the interaction has the screen source. Combining both the non-spatial and spatial domains, the above discussion can be written as RULE 3.11.

RULE 3.11:

a spatial relation is mentioned in the interaction $\Longrightarrow$ the spatial relation probably has the screen source.

We acknowledge that RULE 3.11 is based on intuition rather than hard evidence. Therefore, this rule, like other heuristic rules introduced in this chapter, would be used as the last resource for the resolution when no obligatory restriction can be applied further. In addition, the results drawn from it can always be overwritten in a later stage. In Section §7.5.2, we will talk more about RULE 3.11 in the discussion of the experiment results.

3.11 The heuristics: a revisit

We have presented many heuristics from various resources. They are based on our analysis of regularities appearing in referring expressions with source ambiguities. These rules represent the outcome of our studies that have been carried out so far. Although they are adequate for handling the problems we are facing, they are not and should not be viewed as an exhaustive set. When more complicated source ambiguities come

up, more heuristic rules might be required. Such rules might arise from not only the analysis of regularities, but also the analysis of the user's intentions.

These rules are unconventional in the sense that most of them consider the source of words, which has not appeared in the literature before. However, the ideas behind most of the rules are shared by the natural language processing community. For example, RULE 3.2 to 3.4 use the sources of the surrounding words to help resolving the ambiguities, which is very similar to the local context method for lexical disambiguation [Hearst, 1991]; RULEs 3.5 and 3.6 use the knowledge from conceptual hierarchy for source disambiguation, and this method has been used for sense disambiguation [McRoy, 1992].

Since several heuristics can usually be applied simultaneously to one source ambiguity, an obvious question is what happens when one or more of these rules give contradictory evidence. The solution is to identify priorities among the rules. As the first attempt of resolving source ambiguities, our theory does not offer a firm ground for the priorities, so we adapt a simple approach and give priority to the rules that we feel more confident to use. More than ten rules have been presented in previous sections. They cover various parts of dialogues, from as near as within the same referring expression to as far as several sentences away. Based on both our objective analysis of examples with source ambiguities and our subjective intuition, we classify the rules into two categories: those *about influence within referring expressions* (RULEs 3.1 to 3.4), and those *about influence beyond referring expressions* (RULEs 3.5 to 3.11). The priority is associated with each category in general. A rule about influence within a referring expression is preferred to a rule about influence beyond a referring expression. For example, in principle the evidence from the screen head noun rule is preferred when there is a contradiction between such an evidence and that from the same type rule.

However, we have not found ways to motivate priorities among rules within the same category in a systematic way. But we have implemented a simple and brutal mechanism to handle this problem because such priorities could be required during the actual resolution process. The priorities among the rules depend on the time that the rules are added into the stacks (see Sections §5.4.3 and §6.4.2).

3.12 Summary

In this chapter, we talked about source ambiguities and our approach to handling the problem in more detail. Besides the descriptive content and the referent for each referring expression, we introduced a concept called the *described entity set* which aims at establishing a clear and flexible connection from the descriptive content to the intended referent.

We found a number of linguistic regularities about source aspects within referring expressions and between different referring expressions in the same sentence. These regularities can be used as constraints in resolving source ambiguities.

We argued that the informal definition of a coreferential relation between anaphoric phrases and their antecedents used in previous work is not general enough to cover all possible relations in our system. There could be quasi-coreferential relations between two phrases, which could be used in finding the intended referent of an anaphoric phrase.

The observations in this chapter enable us to design a computational model to resolve source ambiguities. However, before we go into the specific design, we need a meaning representation language capable of representing the meaning being conveyed in two situations, i.e., with and without source ambiguities. The design of this meaning representation language is the topic of Chapter 4.

Chapter 4

The Meaning Representation Language MRL_S

The user's linguistic input, which is in the form of natural language sentences, cannot usually be used directly to access the databases of a system. Intermediate representations are needed to formalise the meaning of the input sentences. Intermediate representations can have several forms, among which a logic-based meaning representation language is widely used. This chapter discusses a meaning representation language called MRL_S (the Meaning Representation Language for Source ambiguities), which is dedicated to the special requirements of sentences with source ambiguities.

4.1 The Design of MRL_S

A meaning representation language is used to express the meaning of an input sentence in a formal form so that it can be processed by the components of a language processing system. Because natural language sentences tend to be vague and ambiguous whereas the language processing components require clearness in the meanings of input, it is not an easy task to build an appropriate meaning representation language. In this section, we present two general considerations of MRL_S. We will briefly review the literature of meaning representation languages with respect to these considerations, and the different properties of MRL_S.

4.1.1 MRL_S as a multilevel language

In the systems we are interested in, the user's input is English sentences. To use them to access the databases, these sentences need to go through processes like parsing, initial semantic processing, referent resolving, and so on. MRL_S is used throughout these processes to represent the meanings. This is shown in Figure 4.1.

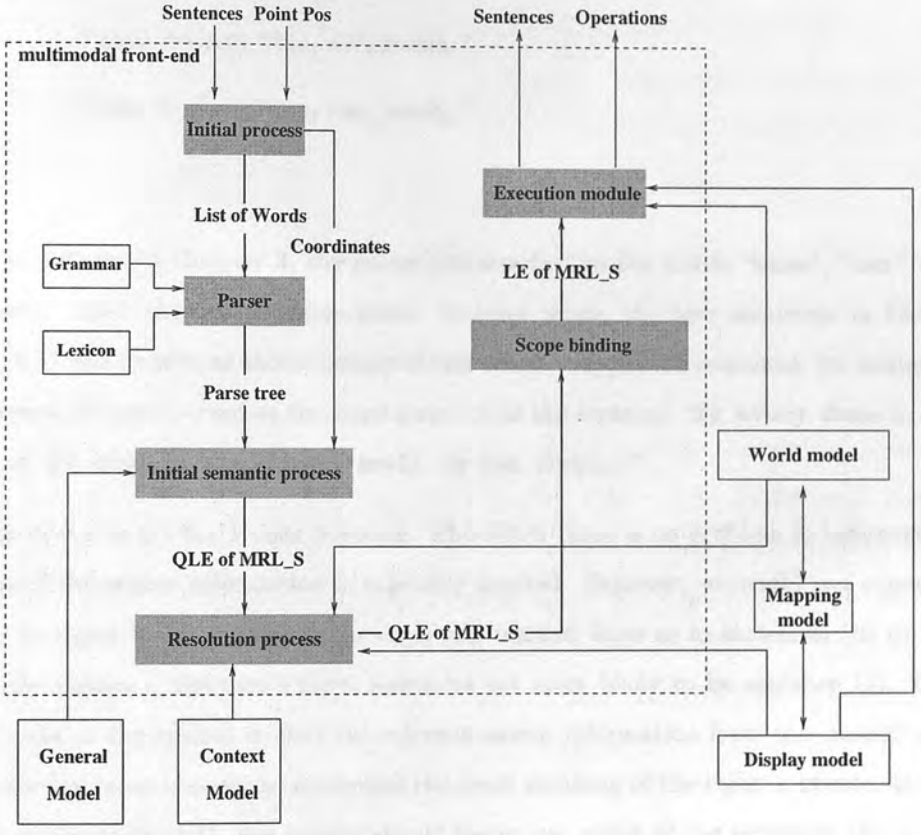


Figure 4.1: MRL_S in a language processing system, such as IMIG

As shown in Figure 4.1, an MRL_S expression for an input sentence is generated from the information produced by the parser. It is decomposed inside the execution module so that parts of it can be sent to the databases of the system to fetch the requested data. Consequently, the characteristics of the input sentences and the databases affect the design of MRL_S. This can be illustrated by sentences in in Example (4.1). Sentence (a) is a normal query in a purely textual question and answer system. However, if the query is to a system where entities and attributes can come from either the domain or

the screen, such as an IMIG system, its meaning can change dramatically. Sentence (a) can be synonymous with each of the four sentences (b) to (e) in Example (4.1).

(4.1) **User:** is every blue car small? (a)

User: Is every blue_d car_d small_d ? (b)

User: Is every blue_d car_d small_s ? (c)

User: Is every blue_s car_d small_s ? (d)

User: Is every blue_s car_d small_d ? (e)

As introduced in Chapter 3, the subscripts attached to the words "`blue`", "`car`" and "`small`" mark the source information. In some sense, the four sentences in Example (4.1) can be seen as abbreviations of four more complicated sentences, for example, sentence (b) can be read as the same meaning as the sentence "`Is every domain car which is blue in the domain small in the domain?`".

Each of (b) to (e) has a clear meaning. Therefore, there is no problem in interpreting them if the source information is explicitly marked. However, we could not expect a user to input his/her query in the explicitly marked form as in sentences (b) to (e). On the contrary, the user's input sentences are more likely to be sentence (a). It is the duty of the system to find the relevant source information from the context and the environment in order to determine the exact meaning of the input sentence. In the case of Example (4.1), the system should figure out which of the sentences (b) to (e) is the intended meaning of (a).

Therefore, the fact that the input sentences usually lack explicit source information requires MRL_S to be able to cope with such sentences at the initial stage of the semantic processing, and to provide enough details for further processing of the sentences. Nevertheless, the source information should be as detailed as that in sentences (b) - (e) of Example (4.1) when the representation of the input sentence reaches the execution module so that the system knows in which database the executions take place. These two different requirements at the two ends of the semantic processing suggest that it would be better to define MRL_S on two levels.

The requirement of multi-level meaning representation language has been addressed in the literature. Allen points out that characteristics of the input sentences are more of a concern of the natural language understanding community, whereas those of databases are more of a concern of researchers on knowledge representation [Allen, 1993]. These two communities have difference in their research goals. People working on natural language understanding are interested in expressiveness and handling ambiguities, whereas those on knowledge representation worry about the modality and complexity of inference processes. This difference causes many current natural language systems connecting a language process component with a knowledge base system to use at least two levels of representations.

Bunt makes the same claim from a different angle [Bunt, 1985]. As a logician, his explanation comes from the design of a model-theoretic semantics for a natural language processing system. Because natural language sentences could have various types of ambiguity, there has to be a semantic representation language L_f ("*a formal language with ambiguous constants*" [Bunt, 1985, P. 115]) to achieve the goal of assigning only one semantic representation to a sentence. However, we also need another formal language L_r whose unambiguous constants that map one-to-one to the objects in the discourse domain (or databases). This is because only this second language can provide a clear "*interpretation of a natural language term in the domain of discourse that is considered*" [Bunt, 1985, P. 115]. Examples of some multi-level meaning representation languages include EFL and MRL of the JANUS system [Weischedel, 1989], and QLF and LF of the CLE system [Alshawi, 1992].

As far as we know, none of the previous multi-level meaning representation languages can handle the meaning representation of sentences (a) and (b) to (e) of Example (4.1) simultaneously without a great amount of revision. Therefore, we designed our own MRL by borrowing ideas from other MRLs, especially that of the CLE system. Our language is called **the MRL for source ambiguities (MRL_S)**.

4.1.2 MRL_S based on predicate logic

Since some processes inside the semantic processing might involve inference, MRL_S is designed in a logical fashion to give as much support as possible. This decision also

provides two other advantages: firstly, a logical form is concise and clear in representation, and secondly, MRL_S in a logical form makes it independent of any particular database format.

To facilitate inferences, most parts of MRL_S are defined in terms of First Order Predicate Logic (FOPL). However, other parts have to go beyond FOPL since a natural language is beyond the expressive ability of FOPL.

MRL_S is not alone in this respect. In fact, most MRLs are based on some forms of predicate logic [Alshawi, 1992, Doe et al., 1992, Androutsopoulos, 1992, Weischedel, 1989]. However, the semantics of MRL_S is not defined by model-theoretic semantics, which is often the case in the literature. One important reason for our decision is that MRL_S is required to represent meanings of commands where the states of affairs before and after the execution of these commands could be different. Commands pose great difficulties on defining logical terms using model-semantics. To avoid putting too much effort in this topic and still achieving correct semantic definition of commands, we choose procedural semantics, which views the meanings of input sentences as procedural calls, as our means for defining the semantics of MRL_S.

In summary, MRL_S has two levels to deal with different requirements imposed at different stages of the semantic processing. At the initial semantic processing stage, the system generates a logical form called a **Quasi Logical Expression (QLE)** from the result of parsing, and at the final semantic processing stage, the system produces another logical form called a **Logical Expression (LE)**. The differences between the QLE and LE will be clarified in the remaining sections.

4.2 Informal descriptions of LE

LE is meant to be free of vagueness and ambiguities like lexical, structural, scoping and referent ambiguities. This is not to say that these ambiguities are beyond our research consideration. On the contrary, source ambiguities, our main research topic, are related to referent ambiguities. We just assume that those ambiguities would be resolved during the process of transforming the input sentence into an LE expression.

In the following discussion, 'LE' could be used in two different ways. Firstly, LE can refer to the final stage language of MRL_S (i.e. MRL_S(LE)), for example, the use of LE in the title of this section. Secondly, LE can stand for the logical expression of a sentence that is represented in the final stage language of MRL_S. Examples of the second use can be found in Section §4.5.1.

4.2.1 LE in command form

Procedural semantics, which was first introduced by Woods in the late 1960s [Woods, 1967, Woods, 1968], has formed the basis of various natural language query interfaces, and still remains influential. It represents the meaning of a natural language sentence as procedural calls to databases.

MRL_S(LE) is a meaning representation language for sentences related to database queries and modifications. To view the meaning of input sentences as procedural calls is a natural and straightforward approach. Additionally, the purpose of MRL_S(LE) is to explicitly annotate the sources of components of the input sentences. Therefore the major difference between procedural semantics and other forms of semantics is not central to our concern.

Similar to another language based on procedural semantics [Fernandes et al., 1994], the topmost representation in LE is a command, which is composed of at least one logical formula and a command type that uses the formula(s) as its argument(s). The command type states the procedure type of the sentence.

As mentioned in Section §1.3.2, we are only considering three types of natural language input sentences. They are yes/no questions, wh-questions and imperative commands. Therefore, LE has three command types, each dedicated to a type of sentence.

4.2.2 LE with source information

One important requirement for LE is to encode explicitly the source information in the logical expression of a sentence. Because all the entities,as well as their attributes and operations, in the systems are divided into those in the screen and those in the domain, variables, constants, predicates, functions and actions in LE all have source

information.

4.2.3 LE with actions

Both of the example domains allow a user to ask questions, and, in addition, to give commands or directly manipulate entities on the screen.

A command or operation is different from a question because the former changes the state of the system (including the display and the databases, etc.), while the latter does not. A command or an operation corresponds to an action in LE, and the database and display are dynamic when considering an action.

Generally speaking, an action could correspond to one simple process in the execution, or a sequence of planned processes that contribute to the same goal. For simplicity, we assume that actions in MRL_S are all simple actions.

So far, we have given a rough idea of LE. In the following sections, we will describe the syntax and semantics of LE more formally.

4.3 The syntax of LE

As a language, LE has several basic blocks, such as numbers, constants and variables. Based on these basic blocks, more complex categories are constructed. In this section, the syntax of LE is presented in a bottom-up manner, so the basic blocks come first.

4.3.1 Constants

In LE, all the logical expressions naming or denoting entities (or classes of entities) in the databases belong to a category called *designator*. As will be shown in the following discussions, the category designator can be divided into even finer classes. For example, a *constant* is a type of designator, which has the form

`cons(`*constant name*, *source*)

In the expression, `cons` is the sign for a constant. Its arguments are a *constant name* and the particular *source* the constant is from. A constant is represented by this complex term as it is necessary to have the name and the source to uniquely identify a constant in our situation.

The possible values of *source* are `screen` and `domain`.

Constants can be classified further into numbers, pairs and individual names.

Numbers

A constant is called a **number** in LE when the constant name is either an integer or a real number, for example `1, 20, 20.5`. At the first glance, it might be surprising to see numbers with sources, but this does happen in a system where mapping relations exist between domain numbers and screen numbers. For example, it is not unusual to have *a human icon on the screen* representing *a specific amount of people in the domain*. In that situation, when a number appears, say 2, it is important to know whether it is a domain number or a screen one. The source information in this case is essential for a full representation of a number. An example of this is `cons(15, domain)`. In some situations, when the source information is clear in some situations, the simplified version `15` can be used.

Pairs

When the constant name is a pair of two integers headed by `pair`, such as `pair(1, 20)`, the constant is called a **pair**. In LE, a pair always represents the coordinates on the screen, so a pair is usually written in the simplified form as `pair(1, 20)`, rather than in the fully specified form `cons(pair(1, 20), screen)`.

Individual names

When the constant name is a string of characters headed by a lower case letter, the constant is an **individual name**. An individual name usually represents an entity from the domain or the screen. The rest of the string can be any letter or digit with or without underscore. For example `cons(car1, domain)` and `cons(icon24,`

`screen)` are two well formed individual names, whereas `pred(is_car, domain)` and `cons(Icon, screen)` are not.

4.3.2 Variables

A *variable* is a logical form representing an entity to be decided in the system. It is also a *designator*. Variables are represented as

`var(`*variable name*, *source*`)`

As with the use of `cons` in the representation of a constant, `var` is the sign for variables. *Variable name* identifies the name of a variable, and *source* marks the source of the variable.

A *variable name* is a string of characters headed by an uppercase letter. For example, `X1`, `Y` and `Abc` are all well formed variable names.

4.3.3 The symbol for corresponding entity relations

One special designator involves mapping relations. It represents the corresponding entity of the entity denoted by another designator acting as its argument. This symbol is needed in the language because the described entities and the intended referent of a referring expression could be the same entity or the corresponding entities via a mapping relation. The individual names and their sources alone in LE cannot represent the relation between corresponding entities. This symbol is introduced in LE to clearly mark the relation (see examples in Section §4.5). The symbol has the form:

`[corresobj(`*source*`)`, *designator*`]`

For example, the expression `[corresobj(domain), cons(car1, domain)]` is a well formed expression for representing the corresponding entity of domain entity **car1**.

4.3.4 Predicate symbols

A *predicate symbol* defines a relation. In our design, besides the common concern about a relation, there is an extra restriction about the source of a relation. The source is

represented here as the source of the predicate symbol. A predicate symbol has the form:

`pred(`*predicate name*, *source*)

`pred` is the sign for the predicate symbol category. A *predicate name* is a string of characters similar to a *constant name.*

Some examples of well formed predicate symbols are `pred(is_car, domain)`, `pred(is_small, screen)` and `pred(is_icon, screen)`.

4.3.5 Propositions

A *proposition* is a chunk of logical form. It can be as simple as a predicate symbol with some arguments, called an *atomic proposition.* In this case, it has the following form, where "#" means that the tokens in the braces should appear at least once

[predicate symbol{, *designator*}$^{\#}$]

In an atomic proposition, source information exists in both the predicate symbol and the designators. As presented in Chapter 3, an entity and its attributes/relations should have the same source. Therefore, a predicate symbol, which is a representation of an attribute or a relation, should have the same source as its arguments, which represent the entities that the attribute or the relation applies to. This can be presented as a principle that every well formed proposition in LE has to satisfy. The principle is called the **source consistency principle**. It is part of the definition of LE language, but is drawn from our analysis of the part of English that we are working on. An example of a well-formed proposition is `[pred(is_small, domain), cons(car1, domain)]`, whereas `[pred(is_small, screen), cons(table1, domain)]` is not because it violates the source consistency principle.

When a proposition is headed by a quantifier, it is called a *quantified proposition.* There are three quantifiers in the current version of LE: the universal quantifier `forall`, the existential quantifier `exists` and the interrogative quantifier `interrog`. Only the first two types of quantifier can be used to lead a quantified proposition. A logical term

containing an interrogative quantifier is not a proposition, but a quantified entity term, which will be discussed in the next section.

A quantified proposition in LE comprises four parts: a quantifier, a quantified variable, and two propositions acting as the restriction and the body, which looks like the following:

`quant(`*quantifier*, *variable*, *restriction*, *body*)

We assume that every new variable in an LE expression is introduced by a quantifier, so there is no free variable in LE (see Section §1.3.2). We also define the scope of a variable as being within the quantified proposition introducing it. If two appearances of a variable name happen in two different quantified propositions, and neither embeds the other, the two appearances should be counted as two different variables.

(4.2)
```
quant(forall, var(X, domain),
              [and, [pred(is_small, domain), var(X, domain)],
                    [pred(is_car, domain), var(X, domain)]
              ],
              [pred(is_expensive, domain), var(X, domain)]
      )
```

For example, (4.2) is a quantified proposition headed by the quantifier `forall`, and the scope of variable `var(X, domain)` is within the proposition.

The third type of proposition is the *complex proposition*. A complex proposition is composed of several propositions, which are joined together by *logical operators*. It has the form

`[`*logical operator*{`,` *proposition*}$^{\#}$`]`.

The logical operators in the current version of LE consist of `not` (negation), `and` (conjunction) and `or` (disjunction). An example for complex propositions is

(4.3)
```
[and, [pred(is_small, domain), cons(car1, domain)],
      [pred(is_car, domain), cons(car1, domain)]
]
```

4.3.6 Quantified designators

Since the input sentences can be wh-questions, and we take the generalised quantifier view that interrogative words are treated as quantifiers in the logical representation of input sentences [Woods, 1987, Alshawi, 1992], there is a group of designators headed by quantifiers that are drawn from interrogative words. These are called **quantified designators**, each of which is composed of an interrogative quantifier, a variable, and two propositions as the restriction and the body.

An example of quantified designators is given in (4.4), where the quantifier `interrog` indicates this.

(4.4)
```
quant(interrog, var(X, domain),
                [and, [pred(is_small, domain), var(X, domain)],
                      [pred(is_car, domain), var(X, domain)]
                ],
                [pred(is_expensive, domain), var(X, domain)]
      )
```

Although the structure of a quantified designator is the same as that of a quantified proposition, the two terms are different. Firstly, the quantifier used in a quantified proposition is an existential quantifier or a universal quantifier, whereas that in a quantified designator is an interrogative quantifier. Secondly, the meaning of a quantified proposition is either `true` or `false`, whereas that of a quantified designator is an entity in the databases.

4.3.7 Actions

An *action* in LE represents an operation mentioned in the user input, which is usually an imperative sentence in our context. It could change the states of the databases of the system. A action in LE is represented as

`[oper(`*action name*, *source*`)`{, *designator*}$^{\#}$`]`.

Considering its source, an action can be either a *domain action* or a *screen action*. The actions we consider are

- adding one/several domain object(s) to the screen (`add`) with the screen source,
- deleting one/several screen object(s) (`delete`) with the screen source, and
- moving one/several object(s) (`move`), which can have the screen or the domain source.

Examples of actions in LE are:

(4.5)
- `[oper(add, screen), [corresobj(domain), cons(car1,domain)]]`
- `[oper(delete, screen), cons(icon1, screen)]`
- `[oper(move, screen), cons(icon1, screen), pair(200,150)]`

4.3.8 Commands

In LE, a logical expression is a command headed by a *command type*. A command type corresponds to a sentence type, where *test* corresponds to a yes/no question, *list* to a wh-question, and *act* to an imperative sentence.

Different logical forms follow different command types. For example, a proposition would follow command type *test*, a designator follows command type *list*, and an action follows the command type *act*. However, all commands share the same form

`[`*command type*, *logical form*`]`.

An example of a well formed command is

(4.6)
```
[test, quant(forall, var(X, screen),
                     [pred(is_icon, screen), var(X, screen)],
                     [pred(is_blue, screen), var(X, screen)]
             )
]
```

4.3.9 Summary

Formally, the language LE is a pair $(\mathcal{L}, \mathcal{R})$ where $\mathcal{L}$ is an alphabet of symbols and $\mathcal{R}$ a set of formation rules that defines what the well formed expressions are.

The alphabet $\mathcal{L}$

The symbols of LE are divided into 12 categories listed below. Some have all their possible values enumerated, so do not need further extension, e.g., the punctuation set and the sources. Some could be extended when LE becomes more powerful, but their current values are enumerated entirely, e.g., quantifiers and commands. The values of the remaining categories are partially listed. The 12 categories are:

1. **Punctuation signs:** the set $\mathcal{T} = \{$`(, ), ,, [, ]`$\}$.
2. **Logical Operators:** the set $\mathcal{LO} = \{$`not, and, or`$\}$.
3. **Quantifiers:** the set $\mathcal{Q} = \{$`forall, exists, interrog`$\}$.
4. **Signs:** the set $\mathcal{G} = \{$`quant, pred, cons, pair, oper, var, func`$\}$.
5. **Command types:** the set $\mathcal{C} = \{$`test, list, act`$\}$.
6. **Sources:** the set $\mathcal{S} = \{$`domain, screen`$\}$.
7. A set $\mathcal{F}$ of **Function names:**.
8. A set $\mathcal{N}$ of **Numbers:**. It includes integers and reals.
9. A set $\mathcal{V}$ of **Variable names:**.
10. a set $\mathcal{I}$ of **individual names.**
11. a set $\mathcal{P}$ of **Predicate names.**
12. a set $\mathcal{A}$ of **Action names.**

The formation rules $\mathcal{R}$

The production rules of LE are given in appendix A in BNF format.

4.4 The semantics of LE

4.4.1 Several assumptions

The semantics of LE is based on the truth value assignment of logical expressions. Before going on to talk about the detail, we first discuss several assumptions used in the definition of truth values.

The truth assignment criteria

The truth value of an LE logical expression is either **True** or **False**, and there is no other truth value. The assignment of a truth value to a logical term is based on the knowledge bases in the system. An LE expression is defined to have the truth value **True** (in short **T**) if it is true for the data in the databases. Its truth value is **False** (in short **F**) if the expression is false for the data in the database. When an LE expression is actually translated from a command or an action, the truth value is **T** if execution of the command or action is successful, and **F** if the execution fails.

Closed world assumption

A database is a finite collection of data, which cannot include everything, even in a particular domain. Therefore, not all of the propositions can find their truth values based on the information in the databases. It is possible that something is "*unknown*" to the databases. The closed world assumption is then used in this situation, which states that whatever is "*unknown*" to the databases has the truth value **F**. As a result, the truth value assignment under the close world assumption is that a proposition has the value **T** if it is supported by the databases, and value **F** otherwise.

Procedural evaluation

Based on Carnap's distinction between *intension* and *extension* [Carnap, 1964], Woods discussed two different methods for evaluating expressions in the LUNAR system [Woods, 1987]. The principal mode of evaluating the language in the LUNAR system, which is based on procedural semantics, is extensional. That is, expressions are

directly evaluated against the database through the procedures of enumerating and checking the individual samples in the databases. This is different to an intensional mode of evaluation, which applies inference rules to other intensional facts. As the MRL_S language is based on procedural semantics, it is more natural to evaluate LE expressions through an extensional mode (i.e. procedures). This procedural view of the meaning of LE expressions also helps us to avoid the rather difficult task of defining the meanings of actions in a declarative way.

Consequently, for each type of LE expression, simple or complex, there is a particular interpretation procedure to accept it as input, and to generate the meaning of the expression. The sign of an expression, such as `cons` for a constant or `oper` for an action, indicates which procedure would be used. Inside a procedure, there are subroutines that interpret the constituents of the expression. The subroutines could be procedures for LE expressions as well.

The following sections will present those procedures in detail. Starting from basic blocks and moving to more complex structures, the presentation is bottom-up in nature.

4.4.2 Designators and DEP

When an LE expression is a designator, a procedure called *Designator Evaluation Procedure (DEP)* would evaluate it. The procedure has the following subroutines:

- If the expression is a constant name (i.e. a number, a pair or an individual name), and has the form `cons(` c, ξ`)` with the sign `cons`, the DEP tries to find an entity that has the name c in the database specified by the source ξ (i.e. the world model if $\xi =$ `domain` or the display model if $\xi =$ `screen`).

- If the expression is a variable `var(`v, ξ`)` where the sign is `var` and v is a variable name in $\mathcal{V}$, the DEP picks up any entity in the database specified by the source ξ to instantiate the variable.

- If the expression is the special designator `[corresobj(` ξ`),`α`]` where the sign is `corresobj`, and α is another designator, the DEP has the following subroutines:

- it calls an DEP as its subroutine to evaluate α into an entity **E1** in the database specified by the source ξ;
- it searches the mapping model to fetch the corresponding entity **E1'** of **E1**;
- it uses **E1'** as the result of the function.

- If the expression is a `quantified designator`, which has the form `quant(interrog, var(`v`,`ξ`),`α`, `β`)`, where the sign is `quant`, the quantifier is `interrog`, v is a variable name, ξ is a source, and α and β are two propositions. The DEP evaluates this expression to a set of entities $\mathcal{E}$. The subroutines of obtaining $\mathcal{E}$ are:

  1. the DEP finds the set $\mathcal{EA}$ whose items are all the entities in the database **DB1** that make the proposition α evaluate to **T**. **DB1** is specified by the source ξ;
  2. it finds the set $\mathcal{EB}$ whose items are all the entities in the database **DB1** that make the proposition β evaluate to **T**.
  3. the set $\mathcal{E}$ is the intersection of the two sets $\mathcal{EA}$ and $\mathcal{EB}$.

4.4.3 Propositions and PEP

The procedure that evaluates a proposition is called *Proposition Evaluation Procedure (PEP)*. It always generates truth value **T** or **F**.

The first step where the PEP evaluates a proposition is to identify whether the proposition is an atomic, a quantified or a complex proposition. When an expression is known to be a proposition, the identification is straightforward and can be achieved by examining the first term of the expression. A proposition is atomic if the first term is the sign `pred`, quantified if the first term is sign `quant` and complex if the first term is a logical operator, such as `and`, `or` and `not`.

The remaining subroutines of PEP are related to the type of a proposition, so they are presented separately as follows.

Atomic propositions

The PEP evaluates an atomic proposition `[pred(`p, ξ`),` $\alpha_1, ..., \alpha_n$`]`, where p is a predicate name, ξ is a source and $\alpha_1, ..., \alpha_n$ are designators, to the truth value **T** if and only if

1. by calling the DEP, all the designators $\alpha_1, ..., \alpha_n$ are evaluated to entities in the database specified by the source ξ;
2. suppose R is a n-ary relation in the database, the PEP finds that there is the relation R existing among the above entities.

Otherwise, the proposition evaluates to **F**.

Quantified Propositions

Suppose $v \in \mathcal{V}$ is a variable name, α and β are two propositions with the source ξ inside them, then the PEP has the following subroutines to evaluate the truth value:

- if the proposition is `quant(forall, var(`X, ξ`),` α, β`)` where the quantifier is `forall`:
  1. the PEP first finds the set H containing all the entities in the database specified by ξ that make α to be evaluated to **T**;
  2. it then finds the set E containing all the entities from the same database that can make β to be evaluated to **T**.
  3. Finally, if H is a subset of E, the PEP evaluates the quantified proposition to **T**, otherwise it evaluates the proposition to **F**.
- if the proposition is `quant(exists, var(`X, ξ`),` α, β`)` where the quantifier is `exists`:
  1. the PEP first find the sets H and E through the same process mentioned in the procedure for a universal quantifier proposition.

2. then if and only if there at least one common entity between the two sets, the PEP evaluates the quantified proposition to **T**, otherwise, it evaluates the proposition to **F**.

Complex propositions

Suppose α_1, ..., α_n are propositions, the PEP evaluates a complex proposition in the following way:

1. when the proposition is `[not,` α_1`]`, the PEP first recursively calls another PEP as its subroutine to evaluate the proposition α_1. If and only if the proposition α_1 is evaluated to **F**, the PEP evaluates the whole proposition to **T**, otherwise, it evaluates the whole proposition to **F**.

2. when the proposition is `[and,` $\alpha_1, ..., \alpha_n$`]`, the PEP recursively calls another PEP as its subroutine for each proposition α_i, $1 \leq i \leq n$,. If and only if all the propositions α_i evaluate to **T**, the PEP evaluates the whole proposition evaluates to **T**, otherwise, it evaluates the whole proposition to **F**.

3. when the proposition is `[or,` $\alpha_1, ..., \alpha_n$`]`, the PEP recursively calls another PEP as its subroutine for each proposition α_i, $1 \leq i \leq n$. If and only if at least one proposition α_i evaluates to **T**, $1 \leq i \leq n$, the PEP evaluates the complex proposition to **T**, otherwise, it evaluates the whole proposition to **F**.

4.4.4 Actions and AEP

An action in LE corresponds to an operation which aims to change the state of databases. Although the result generated from the *Action Execution Procedure (AEP)* is one of the truth values **T** or **F**, its meaning does not represent a truth in the databases, but whether or not the execution of the corresponding operation is successful. The criteria of a successful execution is that all the operations in the procedure have been executed.

Suppose $e_1, e_2, ...e_n$ are designators with source ξ, then the AEP has the following subroutines as its action:

- if the action is `[oper(delete, screen)`, $e_1, e_2, ...e_n$`]`, that is the source ξ is `screen`, then the AEP goes through the following steps:

  1. the DEP is called to evaluate the designator e_i, to an entity $\mathbf{dment}_i$ in the display model, $1 \leq i \leq n$;
  2. all the entities $\mathbf{dment}_i$ and their relevant relations are removed from the display model, $1 \leq i \leq n$;
  3. the mapping representations in the mapping model are checked, and any representation that is not used anymore would be removed from the model.

  If the above procedure succeeds, the truth value **T** is assigned to the action, otherwise **F** is assigned to the action.

- if the action is `[oper(add, screen)`, $e_1, e_2, ...e_n$`]`, that is the source ξ is `screen`, and every e_i is in the form `[corresobj(domain)`, f_i`]`, $1 \leq i \leq n$, then the AEP goes through the following steps:

  1. the DEP is called to evaluate the designator f_i to an entity $\mathbf{wment}_i$ in the world model, $1 \leq i \leq n$;
  2. a screen entity $\mathbf{dment}_i$ is generated in the mapping model as the corresponding entity of the domain entity $\mathbf{wment}_i$, and it is added into the display model, $1 \leq i \leq n$;
  3. during the generation of $\mathbf{dment}_i$, the mapping relations in the mapping model are checked, and any missing mapping relation that is necessary is added to the mapping model;
  4. each designator e_i is defined to be evaluated to the screen entity $\mathbf{dment}_i$, $1 \leq i \leq n$;

  If the above procedure succeeds, the truth value **T** is assigned to the action, otherwise **F** is assigned to the action.

- if the action is `[oper(move,` ξ`)`, e_1, e_2`]`, then the AEP goes through the following steps:

1. the DEP is called to evaluate the designator e_1 to an entity **ent1** in the database specified by the source ξ, and to evaluate e_2 to a spatial position **pos1** in the same database;
2. the AEP checks the area around **pos1** to make sure that there is enough empty space in which to put **ent1**;
3. modify the position record of **ent1** to the new position **pos1**.

If the above procedure succeeds, the truth value **T** is assigned to the action, otherwise **F** is assigned to the action.

4.4.5 Commands and CEP

Similar to actions, commands can have two results, success or failure. Analogous to the notion of truth conditions, these two results are written as **T** for success, and **F** for failure.

The procedure for executing a command is called the *command execution procedure (CEP)*. As a command is the topmost LE expression, the CEP is always the place where the evaluation of an LE expression starts, and from there all the other procedures are called. Therefore, all the other procedures can be seen as subroutines of the CEP.

The first step in the CEP is to check the type of a command. As said, the command type is a sign indicating which process should be used in the evaluation of a query. It also indicates how to interpret the success or failure of the execution of a command. Suppose α is a proposition, ψ is an entity-form, η is an action, then the CEP has the following steps to execute a command:

1. if the command is `[test,` α`]` where the command type is `test`, the CEP calls the PEP to evaluate the proposition α to a truth value. If the result from the PEP is **T**, the CEP assigns **T** to the command, otherwise, it assigns **F** to the command [1].

[1] In fact, we could use another set of values to indicate the result of a command. For example, **S** for the success of a command and **F** for the failure of a command. However, we decide to keep **T** and **F** because of the resemblance between a command and an action and the fact that the result of the latter may have to be computed with the truth values of other expressions. Using **T** and **F** in all types of expressions makes the computation simpler.

2. if the command is `[list, ψ]` where the command type is `list`, the CEP calls the DEP to evaluate the designator ψ to a set of entities in the databases and the DEP returns the set of entities to the CEP for further processing (e.g. displaying on the screen etc.). If the set of the entities from the DEP is not empty, the CEP assigns **T** to the command, otherwise, it assigns **F** to the command.

3. if the command is `[act, η]`, that is the command type is `act`, the CEP calls the AEP to execute the action η. If the result from the AEP is **T**, the CEP assigns **T** to the command, otherwise, it assigns **F** to the command.

4.5 Some Examples of LE

With the syntax and semantics of LE in mind, we now look at some examples of representing sentences in LE.

4.5.1 The representations of the four meanings of Example (4.1)

At the beginning of this chapter, we gave four different possible meanings for the sentence "`is every blue car small?`". With the expressive power of LE, especially the source annotation, these four meanings can be easily distinguished by their LE expressions.

Meaning 1: is every blue$_d$ car$_d$ small$_d$?

The command type in the LE expression of this sentence is `test` because the sentence is a yes-no question. The subject is a noun phrase with a universal quantifier, so its corresponding LE expression is a quantified proposition whose quantifier is `forall`. The variable of the quantified proposition is a domain variable (i.e. `var(X, domain)`). This means that the intended referent of the subject, which is represented by the variable, is a domain entity. In summary, the LE expression for this meaning is:

(4.7)
```
[test,
    quant(forall, var(X, domain),
        [and, [pred(is_blue, domain), var(X, domain)],
              [pred(is_car, domain), var(X, domain)]
```

```
                ],
                [pred(is_small, domain), var(X, domain)]
            )
        ]
```

Meaning 2: is every blue$_s$ car$_d$ small$_d$?

The command type of the LE expression for meaning 2 should be the same as that for meaning 1. The predicative adjective is still a domain word, so the LE expression for it is kept the same, and so is the LE expression for the intended referent of the subject (i.e. the variable). However, this time the subject noun phrase is a mixed-source phrase, where the source of the adjective "`blue`" is the screen. Therefore, the argument of the predicate symbol `is_blue` should not be `var(X, domain)` directly, but the corresponding entity of the variable, which is `[corresobj(domain), var(X, domain)]` in LE. In summary, the LE expression for this meaning is:

(4.8)
```
[test,
    quant(forall, var(X, domain),
        [and, [pred(is_blue, screen),
                    [corresobj(domain), var(X, domain)]
              ],
              [pred(is_car, domain), var(X, domain)]
        ],
        [pred(is_small, domain), var(X, domain)]
    )
]
```

Meaning 3: is every blue$_s$ car$_d$ small$_s$?

Different from that in meaning 2, the predicative adjective in this meaning becomes a screen word. Accordingly, the intended referent of the subject is a screen entity, which can be written as `var(X, screen)`. This time, it is not the predicate symbol `is_blue` but the symbol `is_car` that has different source from the variable, so the symbol `corresobj` is added in front of `is_car` to make the term satisfy the source consistency principle. Note that this `corresobj` has a different source from the one in (4.8). The LE expression for this meaning is:

(4.9)
```
[test,
```

```
            quant(forall, var(X, screen),
                [and, [pred(is_blue, screen), var(X, screen)],
                      [pred(is_car, domain),
                           [corresobj(screen), var(X, screen)]
                      ]
                ],
                [pred(is_small, screen), var(X, screen)]
            )
        ]
```

Meaning 4: is every blue$_d$ car$_d$ small$_s$?

Similar to meaning 3, the predicative adjective in this meaning is a screen word, but this time, both words in the subject are from the domain, so the function `corresobj` is used in front of the two predicates of the two words. The LE expression for this meaning is:

(4.10)
```
        [test,
            quant(forall, var(X, screen),
                [and, [pred(is_blue, domain),
                           [corresobj(screen), var(X, screen)]
                      ],
                      [pred(is_car, domain),
                           [corresobj(screen), var(X, screen)]
                      ]
                ],
                [pred(is_small, screen), var(X, screen)]
            )
        ]
```

4.5.2 Other LE examples

The above examples illustrate some basic features of LE, but they only cover a very small fraction of the linguistic phenomena that LE can handle. The following examples demonstrate how LE is used for representing proper names, wh-questions and imperative sentences.

Proper names

Proper names are represented as constants in LE. For example, the proper name "icon1" is represented as `cons(icon1, screen)`, where the source is explicitly marked. Suppose the input sentence is "`which blue`$_d$ `car below icon1 is expensive`$_d$?", its LE expression is:

(4.11)
```
[list,
    quant(interrog, var(X,domain),
        [and, [pred(is_blue, domain), var(X, domain)],
              [pred(is_car,domain), var(X, domain)],
              [pred(is_below, screen),
                   [corresobj(domain), var(X, domain)],
                   cons(icon1, screen)
              ]
        ],
        [pred(is_expensive, domain), var(X, domain)]
    )
]
```

Wh-questions

A wh-question has the command type `list`. This command type distinguishes it from a yes-no question, although the subject of a wh-question is also a quantified term headed by the quantifier `interrog`. (4.11) is an example of an LE expression for a wh-question.

Imperative sentences

An imperative sentence has the command type `act`. If the subject of the imperative sentence is missing, the system would add appropriate signs to explicitly mark the subject so that it is clear what would carry out the action. For example, if the user asks the system to delete an icon by entering the command "`delete icon1`", a subject sign `cons(imig,screen)` is added, which makes the LE expression of this sentence look like this:

(4.12)
```
[act,
    [oper(delete, screen), cons(imig, screen),
```

```
                                    cons(icon1, screen)
        ]
    ]
```

The subject addition mentioned above does not aim at the current application coverage since all the actions are performed by the system. Its benefit lies in the complete structure of LE expressions for actions so that any future extension to include actions from the user or other participants would be straightforward.

4.6 The Design of QLE of MRL_S

QLE (Quasi Logical Expression) is the logical form of MRL_S at the initial stage of the semantic processing. A QLE expression is generated from the parsing result, whereas the corresponding LE(s) could not usually be generated until the end of the semantic processing. Between these two stages, source ambiguities and other sorts of ambiguities in the input sentences might have to be resolved. According to Alshawi et al's experience with the CLE, the approach of separating the resolution of some types of ambiguities from that of others "*can effectively reduce the complexity of the system as a whole, and avoids multiplying out interpretation possibilities at an early stage, considerations which are important to achieving wide coverage in a natural language processing system.*" [Alshawi, 1992]. We adopted their approach, so the transformation from QLE to LE is a gradual process where some QLE terms are removed and the corresponding LE terms are added by those processes.

Under this scheme, QLE is obviously a superset of LE, so QLE shares the same semantic approach as LE, that is, all its logical expressions are in command form. Actually, an LE expression inherits its command type from the corresponding QLE, and the QLE expression obtains its type from the syntactic structure of the input sentence.

In our design, the meaning representation language is only required to clearly represent the meaning of input sentence with available information. It does not have disambiguation ability. Resolving ambiguities is the task of other components. Therefore, no knowledge base checking is performed when translating the parsing results into QLE expressions.

4.6.1 Main extensions of QLE

QLE is required to represent correctly and concisely sentence meanings with probable source ambiguities, and at the same time to provide as much support as possible to inference in the language understanding process.

The main extensions of QLE from LE are in the representation of unresolved sources, references and unscoped quantifiers. The idea of constructing QLE based on LE is inspired by Alshawi and his colleagues' work on QLF in CLE [Alshawi, 1992]. The special terms only used by QLE only are:

1. Some linguistic information in a noun phrase may be important in the ambiguity resolving process. It is essential to keep it in the QLE expression so that it can be used whenever it is needed. The term *category* is such a place in QLE for those linguistic attributes. A category term is a list of feature-value pairs in the form $[feature_1 : value_1, \ldots, feature_n : value_n]$.

   The definitions of features here resemble those in the CLE [Alshawi, 1992] as the utilities of these features in MRL_S and the CLE are similar. Those features and their possible values are:

   - `typ` for expression types, whose values are `quant` for quantified phrases and `ref` for noun phrases;
   - `phr` for phrase types, whose values are `ana` for anaphoric phrases, `def` for definite descriptions and `dex` for pointing phrases;
   - `quant` for quantifier types, whose values are `exists` for various existential quantifiers, `forall` for various universal quantifiers and `interrog` for quantifiers from wh-phrases.
   - `lex` for the surface form of phrases;
   - `num` for number information, whose values are `sing` for singular, and `plur` for plural.

   For example, the category terms of noun phrases "`the blue car`", "`every small car`" and "`it`" are

`[typ:ref, phr:def, num:sing, lex:the blue car]`

`[typ:quant, phr:def, num:sing, quant:forall, lex:every small car]`

`[typ:ref, phr:ana, num:sing, per:third, lex:it]`

2. *A term called* `qpred` *for a source undetermined predicate symbol* and *a term called* `qvar` *for a source undetermined variable.* These terms correspond to the types of entity terms in QLE which have source ambiguities. As presented in Chapter 3, proper names are assumed not to have source ambiguities.

3. *A term called* `qterm` *for an unscoped quantified expression or an undetermined definite description.* This term is used to provide a vague translation for a quantifier at the pre-scoping stage or for a definite description/pronoun whose referent has not been decided yet. A qterm has the form

   `qterm(`*category, variable, restriction*`)`

   Category has been presented in above, and *Variable* in QLE is in `qvar` form. *Restriction* is a QLE proposition.

At the time a QLE expression is constructed from the parsing results, the scope of a quantifier, if there is one, is unidentified, which bring the `qterm` term in QLE. Because of the structural difference between `qterm` and `quant`, the positions of predicates and quantifiers in a QLE expression are different to those in an LE expression. In particular, a quantifier in QLE is located at the restriction part of another quantifier rather than at the body part like that in LE. This can be seen from (4.13).

4.6.2 Examples

The following are some examples of QLE expressions.

Example 1

In Section §4.5, we gave the four different LE expressions for the sentence "`is every blue car small?`" These four meanings share the same QLE, which is:

(4.13)

```
[test,
    [qpred(small),
     qterm([typ:ref, phr:quant, num:sing, lex:every blue car],
          qvar(X), [and, [qpred(is_blue), qvar(X)],
                         [qpred(is_car), qvar(X)]
                   ]
          )
    ]
]
```

In the logic form, several special QLE terms are generated due to the lack of source information, such as `qpred` and `qvar`.

Example 2

If there is a pronoun in the input sentence, the category term is essential for the resolution of the pronoun in the later stage, whereas the restriction part is usually empty. For example the QLE expression for the sentence "`is it cheap?`" is:

(4.14)

```
[test,
    [qpred(cheap),
     qterm([typ:ref, phr:ana, num:sing, per:third,
            gen:neutral, lex:it
           ],
           qvar(X), []
          )
    ]
]
```

4.7 Summary

In this chapter, we talked about the meaning representation language MRL_S, which is dedicated to handle the meaning of sentence with source ambiguities. Because of the complexity of the meaning representation task, the language has two parts. The QLE part is designed to represent the sentence meaning where there are many unsolved ambiguities, whereas the LE part is developed to handle various source information in the logical expressions, such that the execution model knows where to evaluate the logical terms.

The discussion of LE was the main topic of this chapter. Both formal and informal descriptions of the syntax and the semantics of LE were given in order to achieve a clear definition of the MRL. This was followed by some examples, which showed how sentences are represented in LE expressions.

QLE is a superset of LE due to the complexity of resolving various ambiguities. Extensions in QLE are used to represent those logical terms whose scope, referent or source information is missing. Some examples are also given to show how sentences are represented in QLE expressions.

Based on this meaning representation language, we will talk about the method used for resolving source ambiguities inside a sentence in Chapter 5.

Chapter 5

Source and Reference Evaluation as a CSP

In this chapter, we will talk about our approach to resolving source ambiguities and references. Because the resolution of source ambiguities and references are interrelated together, we have designed a process to handle both cases simultaneously. The central idea of the process is to treat the resolution as a constraint satisfaction problem, in which the variables and constraints are formalised from various origins.

5.1 A brief summary of source ambiguities and referring expressions

The resolution process to be presented in this chapter aims at resolving source ambiguities and some other referential ambiguities. The knowledge and rules used in the process have been presented in Chapter 3. However, they are briefly summarised in this section in order to help the reader to understand the mechanisms used in the resolution process.

A **source** is a tag adding to an entity or a feature (i.e. an attribute or a relation) to mark the place the entity belongs to. It must be the domain or the screen. A **source ambiguity** means that the source information is vague. To handle the reference disorientation problem in a systematic and uniform way, we introduced the **described entity set**(DES) for a referring expression, in addition to the descriptive content and

the **intended referent**. The described entity set is an intermediate representation in the course of finding the intended referent for each referring expression. However, as we emphasised in previous chapters, this does not mean that the described entity set should be found first in any given situation. We just found that it would be much easier to find the intended referent if the described entity set had been found. Consequently, from the view of a conceptual and systematic approach, each referring expression has a described entity set between its descriptive content and its intended referent. Whether or not the described entity set is found before finding the intended referent depends on the actual resolving situation.

The intended referent is a single entity when the referring expression is singular, whereas the described entity set in the same situation is a set of entities. All the entities in the set are described by the descriptive content of the phrase. The word **describe** means that each of these entities satisfies at least one component of the descriptive content. The intended referent is either the same as, or the corresponding entity of, an element of the described entity set.

Several linguistic regularities related to the sources of the words in referring expressions were presented in chapter 3. They could be used as restrictions on determining the sources of words. They are enumerated again here:

- Restrictions inside referring expressions
  - RULE 3.1: The deictic element of a phrase is a demonstrative word with a pointing action $\Longrightarrow$ the screen entity that is pointed to must be in the described entity set.
  - RULE 3.2: The head noun of a phrase unambiguously names a screen category $\Longrightarrow$ the intended referent of the phrase must be a screen entity.
  - RULE 3.3: The head noun of a phrase unambiguously names a screen category $\Longrightarrow$ the described entities described by the non-classifier modifiers of the phrase must be screen entities.
  - RULE 3.4: **[Screen head noun rule]** The head noun of a phrase unambiguously names a screen category $\Longrightarrow$ the intended referent of the phrase

must be a screen entity and the described entities described by the non-classifier modifiers of the phrase must be screen entities as well.

- Intra-sentential source influence among referring expressions
  - RULE 3.5: **[Same type rule]**two words in the same sentence share the same semantic type $\Longrightarrow$ they probably have the same source.
  - RULE 3.6: **[Same position rule]** two words are at the same position in two phrases, and the two phrases have the same structure $\Longrightarrow$ the two words probably have the same source.
- Inter-sentential source influence
  - RULE 3.7: a word appears more than once in adjacent sentences and all of the appearances are generated by the same user $\Longrightarrow$ all these instances of the word probably have the same source.
- RULE 3.8: For an anaphoric phrase and its antecedent:
  - The head nouns of the two phrases are the same, or one is a generalisation/specification of the other, or they have a mapping relation between them.
  - Suppose set A contains attributes represented by pre-modifiers of the anaphoric phrase, set B contains the attributes represented by pre-modifiers of the antecedent, set C contains the attributes that have mapping relations with those in set A, and set D contains attributes that have mapping relations with those in set B, then the union of A and C is a subset of the union of B and D.
- Salience influence
  - RULE 3.9: When selecting from several discourse entities that are available to be the intended referent of a referring expressions, the one that has highest salience is preferred.
- Spatial position influence

- RULE 3.10: When selecting from several screen entities that are available to be the intended referent of a referring expressions, the one that is nearest to the pointing position is preferred.

- Domain Specific source influence

  - RULE 3.11: a spatial relation is mentioned in the interaction $\Longrightarrow$ the spatial relation probably has the screen source.

When the current referring expression is an anaphoric one, for example, a definite noun phrase in anaphoric use or a pronoun, the coreferential relation between this phrase and its antecedent is traditionally seen as crucial to referent evaluation. In the processing of referring expressions with source ambiguities, the coreferential relation is defined between two phrases with the same intended referent and a new relation, called **quasi-coreference**, is defined to cover more complicated cases, i.e., the intended referents of two phrases are the entities corresponding to each other.

5.2 Basic concepts of a constraint satisfaction problem

The theme of this chapter is the issues of formalising reference evaluation and source disambiguation as an integrated constraint satisfaction problem (CSP). To facilitate readers from various background, we first sketch some basic CSP materials that are used either in our discussion, such as concepts about variables and constraints, or in the actual resolution process, such as the AC-3 algorithm. These materials are mainly from papers by Mackworth, Mellish, Kumar, Rich and Knight, and others [Mackworth, 1977, Mellish, 1985, Haddock, 1988, Rich and Knight, 1991, Kumar, 1992].

5.2.1 A constraint satisfaction problem

Constraint satisfaction, as a general problem solving method, has long been of interest in Artificial Intelligence and Computer Science research [Mackworth, 1977, Mellish, 1985, Nadel, 1985, Haddock, 1988, Kumar, 1992]. Informally, presented by Kumar [Kumar, 1992], a **Constraint Satisfaction Problem** (CSP), at least the ones we are interested in, consists of:

- a set of *Variables*,
- a finite and discrete *domain* of *possible values* associated with each variable, and
- a set of *constraints*, each of which defines some subset of the original set of variables and limits the combinations of values that the variables in this subset can take.

The *goal* is to find a set of assignments to all the variables such that the set of assignments satisfies all the constraints.

If a real world problem has several concrete outcomes to achieve, and there are restrictions on and between the features of each individual outcome, the problem can be formalised as a CSP.

5.2.2 Variables

During the formalisation, the outcomes to be achieved are represented as **variables**. A possible value of a variable x is called a **candidate** of x, and the set that contains all the candidates of x is defined as the **domain** of x, which can be written as D_x. The candidate that appears in the result assignment of x is called the **solution** of x. The **assignment** of candidate v to variable x can be written as $< x, v >$. If there are several simultaneous assignments of candidates $v_1, v_2, ..., v_n$ to a set of variables $x_1, x_2, ..., x_n$, they are defined together as **a compound assignment**, which can be denoted as $(< x_1, v_1 >, < x_2, v_2 >, ..., < x_n, v_n >)$.

In a CSP, a variable can be:

1. *unresolved*: all the candidates in its domain are possible solutions;
2. *partially resolved*: its candidates has been limited to several, but the solution has not been identified yet; or
3. *fully resolved*: the solution has been found, i.e., the remaining candidates are the solution.

Because the variables usually are resolved one by one in a CSP, all the above three states can exist across different variables at the same time. This imposes requirements on the representation of variables. In order to resolve these variables in a CSP, on the one hand, there should be no fundamental difference (at least superficially) between the representations of variables in any of the three states so that the transformation from one state to another is not a barrier, on the other hand, there has to be some difference in the representations for the purpose of identification.

In order to have these variables resolved in a CSP, on the one hand, the representations of variables in any of the three states should be no fundamental difference (at least superficially), on the other hand, some identification should be available in the representation to differentiate variables in a state from those that in the other two states.

5.2.3 Constraints

The restrictions on the features of a variable or the relations between/among variables are formalised as **constraints**. Formally, a n-ary constraint C on a set of variables $x_1, x_2, ..., x_n$ can be written in a predicate form $C(x_1, x_2, ..., x_n)$.

When an abstracted problem is simple, the constraints are usually explicitly expressed in its description. However, this is seldom the case in a real application. It is more common for the constraints to be hidden behind all sorts of other information. Therefore, there has to be some criterion that determines what can be considered a constraint. Generally, people think that all constraints share two important features: *easily comprehensible* and *locally computable* [Rich and Knight, 1991].

There are two stages in using CSP to solve a problem. Firstly, the problem has to be represented into a constraint satisfaction problem. That is, variables and constraints have to be formalised from the original problem. This is called the stage of *formalisation*. Then, secondly, the constraints are evaluated against the variables to find the solutions for the variables. This is called the stage of *constraint satisfaction*.

Constraints can be formalised differently. They can be generated by knowledge engineers when the CSP system is built. In this case, the CSP system cannot change the

constraints generated by human, which could be too restrictive in some applications. Constraints can also be generated by the CSP system when a concrete problem arrives, such as resolving the ambiguities in an input sentence. This method provides much flexibility. In the latter case, constraints can be formalised merely during the formalisation process. Once they are generated and sent to the constraint satisfaction process, they cannot be changed, and no new constraint would be formalised. The other way is that constraints can be formalised in both the formalisation and the constraint satisfaction processes. That is, new constraints can be generated when the existing constraints are applied to restrict variables. Since a CSP system capable of generating constraints in the last way is much more complicated than others, it is seldom used unless the problem requires it.

Constraints can be divided into different types, according to the number of arguments. A **unary constraint** only has one variable, that is, it represents a feature of one variable, for instance $car(x)$. A **binary** has two variables and it concerns the relation between its variables. There could also be **higher order constraints** that involve relations among even more variables.

The effects of constraints are of two types. They can be obligatory so that any candidate that does not satisfy them is rejected. They can also be optional, so some candidates are picked up preferentially, but this choice can be overridden later.

A compound assignment $(< x_1, v_1 >, < x_2, v_2 >, ..., < x_n, v_n >)$ is said to **satisfy** a constraint C if and only if the variables in the compound assignment are the same as the variables of the constraint C, and $C(v_1, v_2, .., v_n) = True$. It can be written as $satisfy((< x_1, v_1 >, < x_2, v_2 >, ..., < x_n, v_n >), C(x_1, x_2, .., x_n))$ iff $C(v_1, v_2, .., v_n) = True$.

5.2.4 Propagation and search

As Rich and Knight stated [Rich and Knight, 1991, pp. 89-90], two sub-processes can happen during the satisfaction of constraints.

The first is **propagation**, which is a process of trying to achieve consistency in the network. Various levels of consistency can be achieved. The higher the level of consistency,

the better the chance to converge to a solution. However, as the level of consistency increases, the complexity of the algorithm and the structural complexity of the network increase dramatically. So there is a trade off between the level of consistency and the complexities.

Propagation terminates for one of two reasons. The first is when the candidate set of a variable becomes empty, which means that an **inconsistency** has happened. In this case, a sub-process called **backtracking** is initiated to get rid of the inconsistency. If the inconsistency cannot be removed, the satisfaction process fails. The second reason is when there is no more change happening in the candidate sets of variables. If a solution has been found, the satisfaction succeeds and stops. Otherwise, the second sub-process of constraint satisfaction will be used.

The second sub-process is called **search**, which chooses a particular variable and one of the candidates according to some arrangement (such as using a preference), so that the propagation process can proceed from the new state. The selection of a specific variable and value can vary in different problems and approaches.

In the last two decades, a number of constraint-satisfaction algorithms have emerged. They range from simple ones like *generate and test*, and *simple backtracking* to more complicated ones like *full lookahead* and *really full lookahead*. Many heuristics have been developed for selecting variables and candidates (see Kumar's survey [Kumar, 1992] for more detail).

5.2.5 Constraint networks and network consistency

When a CSP only contains unary and binary constraints, it is a **binary CSP**. A binary CSP can be represented naturally as a graph called a **constraint network**. In a constraint network, variables are nodes, unary constraints are loop arcs that start from and terminate in the same node, and binary constraints are links between two nodes. Constraint networks are studied extensively by researchers like Mackworth [Mackworth, 1977]. Mackworth's network consistency approach [Mackworth, 1977] aims at achieving constraint satisfaction through achieving consistency in the network. It works at a local level, monotonically removing impossibilities step by step. Because

the method is simple and has a good cost-effect ratio, it was used by Mellish and Haddock in their CSP approaches to reference evaluation [Mellish, 1985, Haddock, 1988].

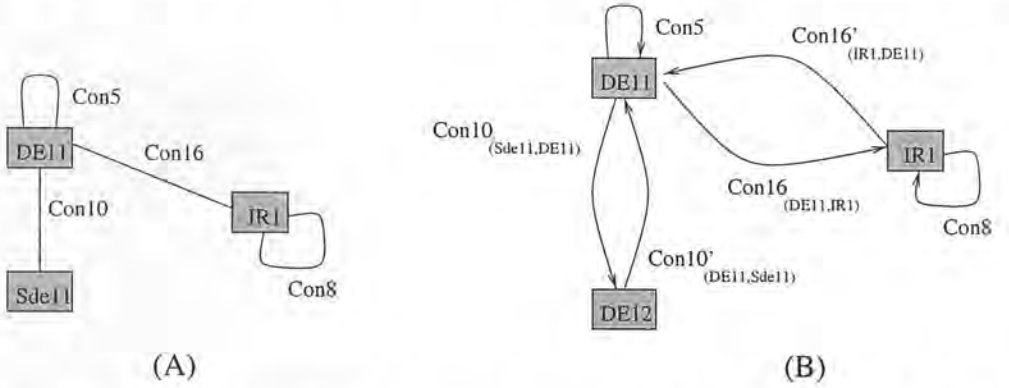


Figure 5.1: The transformation from a non-directional graph (A) to a directional graph (B)

In a constraint network, a node is **node consistent** if and only if every candidate in the candidate set of the node satisfies its unary constraints. If every node in a network satisfies the node consistency, the network is node consistent. Node consistency can be achieved by checking the network thoroughly. After a network becomes node consistent, it will always hold this consistency.

Another type of consistency is arc consistency. Arc consistency occurs between two nodes and it is *directional*. An arc from node x to node y being arc consistent does not imply that the arc from node y to node x is consistent. When considering arc consistency, a constraint network is better viewed as a directional graph. It is straightforward to transform a non-directional graph to a directional one. An example is given in Figure 5.1.

An arc C from node x to y is **arc consistent** if and only if for $\forall i \in D_x$, $\exists j \in D_y$ such that $satisfy(< x, i >, < y, j >, C_{x,y}) = C(i,j) = True$. The *REVISE* process shown below achieves arc consistency for an arc $C_{x,y}$ and returns an flag indicating whether or not some candidates have been removed.

procedure REVISE($C_{x,y}$):

1. set the value of $DELETE$ to $false$, which means that there is no

change in the domain of node x (D_x).

2. if all the candidates in D_x have been selected, return the value of $DELETE$.
3. select a candidate i from D_x.
4. check each candidate j in the domain of node y (D_y), if none of them can make $C_{x,y}(i,j)$ to be *true*, delete i from D_x, and set the value of $DELETE$ to *true*.
5. go back to step 2.

Arc consistency is different from node consistency in that it may not always hold after being achieved. Arc $C_{x,y}$ may be consistent at one point, but may become inconsistent when some arc $C1_{y,z}$ is subsequently revised. Then the question is when and which arcs could be affected.

Mackworth [Mackworth, 1977, p. 105] presented a rule, which says that when the candidate set of variable x is changed due to achieving arc consistency of $C_{x,y}$, the consistency status of all the arcs $C_{z,x}$ whose end point is x where $z \neq x, z \neq y$ may be affected and therefore need to be checked again.

When a network has all its nodes node consistent and all its arcs arc consistent, it is said to be in **network consistency**. Mackworth's AC-3 is a good algorithm to achieve network consistency for a constraint network.

procedure AC-3:

1. achieve node consistency for each node x of the network.
2. build a set Q that contains all the arcs $C_{x,y}$ of the network.
3. if Q is empty, stop.
4. select and remove an arc $C_{x,y}$ from set Q.
5. call $REVISE(C_{x,y})$ to achieve arc consistency for arc $C_{x,y}$.
6. if the return value of $REVISE(C_{x,y})$ is *true*, then add all the arcs $C_{z,x}$ whose end node is x into Q, where $z \neq x$ and $z \neq y$.

7. go back to step 3.

In a binary CSP where only node and arc consistency is required, constraint propagation happens in the process of achieving arc consistency. The change to the candidates of one variable may affect the candidates of another, thus providing propagation.

The AC-3 algorithm is the one we used in the resolution process to achieve node and arc consistency. Because it works on local information and monotonically removes impossibilities, it provides a mechanism that is simple and has good cost-effect ratio for our source and reference disambiguation.

Now, we have refreshed the memory of the results presented in Chapter 3 and the background materials about CSP. In the following, we will move to the discussion of the main theme of this chapter, representing reference evaluation and source disambiguation as a CSP.

5.3 Reference evaluation and source disambiguation as a CSP

Two often mentioned motivations of applying constraint satisfaction method to a problem are the facts that the problem can be represented into a constraint satisfaction problem and the resolving of the problem can benefit from the representation. In this section, we will argue that reference evaluation and source disambiguation are two such problems that can be represented into constraint satisfaction problems, and their resolution process would be benefited by doing so.

5.3.1 Reference evaluation (RE)

Reference evaluation is a process of finding the entity referred to by a noun phrase. It should only be used when a noun phrase requires a referent. Among the types of noun phrases we consider, only definite noun phrases, deictic phrases and pronouns require the reference evaluation process. The other two types of phrases (indefinite noun phrases and quantified phrases) seldom need the process in a question-answering system like IMIG, although they might need it in systems that allow declarative sentences, e.g.

[Mellish, 1985]. For instance, in (5.1), the indefinite noun phrase "`a blue car`" does not refer to any specific entity, because it can match any entity which satisfies its description.

(5.1) "`Do you have a blue car?`"

An interrogative phrase could be assigned a particular entity after the execution against the databases succeeds, but this entity is not to be identified during semantic processing, where reference evaluation happens.

As a result, we will only consider definite noun phrases, deictic phrases and pronouns in the reference evaluation process.

All these phrases refer to entities in the knowledge bases of the system. The knowledge bases can be either a representation of the hearer's memory, which keeps track of the salient entities mentioned by linguistic utterances or manipulated by gestures, or a model of the hearer's visual perceptual world, which stores all the entities that can be seen by the hearer. Mapping into the databases of the IMIG system, the hearer's memory and visual perceptual world are represented as the context model and the display model respectively.

Treating reference evaluation as a CSP was first motivated by Mellish [Mellish, 1985], and then by Haddock [Haddock, 1988]. In his book about interpreting natural language descriptions, Mellish presented detailed discussions and algorithms. He identified the variables and constraints of the CSP for reference evaluation as [Mellish, 1985, p. 42]:

> *"Reference evaluation is the task of deciding which object(s) in the world a given phrase refers to, and can be seen as the process of instantiating a variable with an appropriate value. Many factors influence what possible instantiations can be considered — these act as constraints on the value of the referent."*

5.3.2 Source disambiguation (SD)

Source disambiguation is a process of finding the appropriate source of each word that needs a source. In our system, the word categories that need a source are verbs, nouns,

adjectives and prepositions representing spatial relations.

To identify the source of a word, one needs to consider the characteristics of the attribute given by the word. Some attributes only belong to one type of entity in one source, for example, the attribute `being a car` only belongs to those entities that are cars in the domain. Therefore, whenever this attribute appears, its source is the domain, as is the source of corresponding word "`car`".

Not all attributes act as consistently. Some attributes can belong to one type of entity in one database in one situation, but to another type of entity in another database in a different situation. Consequently, before the information related to the source of an attribute (e.g. the entity, the situation, and so on) is available, it is very difficult to identify the source.

This indicates that the entity that the attribute belongs to provides crucial evidence relating to the correctness of some source assignments. If a word to be resolved is in a referring expression whose referent is to be found in reference evaluation, the results of evaluation could be used as checking criteria.

What if a word that needs a source does not belong to a referring expression, for example, a word in an indefinite phrase or an interrogative phrase? An indefinite phrase describes various characteristics of a group of entities. As mentioned in Section §1.3.2, we assume that at least one entity in the system would satisfy the description of such a phrase. The same assumption also holds for an interrogative phrase. Therefore, the correctness of a source assignment to a word in these phrases depends on the fact that at least one entity is found in the system having the attribute represented by the word with the source assigned[1].

Similar to reference evaluation, the source disambiguation process can also be treated as a CSP. There are many restrictions on the selection of the source value (such as those rules we mentioned in Chapter 3 and summarised in Section §5.1), and they naturally act as the constraints on the words, the variables.

[1] We acknowledge that this assumption is probably too strong since it means that the system cannot interpret a query like "`is there a red`$_d$ `car`$_d$" when there is no red car in the databases.

5.3.3 RE and SD as an integrated CSP

We have argued that both reference evaluation and source disambiguation can be viewed as constraint satisfaction problems. In addition, the following is the evidence that these two processes are better integrated into one constraint satisfaction process.

In Section §5.3.2, we mentioned that source disambiguation needs the results from reference evaluation to verify its source assignments. In fact, as we will show, the performance of a reference evaluation process also depends on source disambiguation.

Traditionally, people follow Frege's idea that each referring expression has a sense and a referent, and it is a principle that the reference is determined by the sense [Frege, 1975]. Kronfeld gives an informal definition of the sense, although he uses the name *descriptive content* instead. He thinks that the descriptive content of a phrase is "*a function of the meanings of the words that appear in that description*" [Kronfeld, 1990, p. 23]. It also makes sense to say that the meanings of the words inside a phrase provide critical information for finding the referent.

Unfortunately, the existence of source ambiguities in the words in a phrase make these words ambiguous in meaning, and therefore affect the descriptive content of the whole phrase. Since the referent is decided by the sense (the descriptive content), the referent could be ambiguous even without any other form of referential ambiguity. In this situation, resolving source ambiguities among the words becomes a precondition of finding the correct referent for a referring expression.

This evidence drives us to view the source disambiguation and reference evaluation processes as an integrated constraint satisfaction problem.

Let's look at an example. Suppose (5.2) is an input command. In order to interpret it correctly, the sources of the words "`blue`" and "`remove`" have to be identified. In addition, each of the two referring expressions "`the blue car`$_d$" and "`the screen`$_s$" requires a described entity set and an intended referent. There can be many relations between the words and the entities. A few obvious ones are listed below:

(5.2) "`remove the blue car from the screen.`"

1. The source of the word "`remove`" is the same as that of the intended referent of the phrase "`the blue car`";
2. The source of the word "`remove`" is the same as that of the intended referent of the phrase "`the screen`";
3. One described entity in the DES of the phrase "`the blue car`" must be a car;
4. One described entity in the DES of the phrase "`the blue car`" must be blue;
5. The described entity in the DES of the phrase "`the screen`" must be the screen.
6. The intended referent of the phrase "`the blue car`" must be removable;
7. The intended referent of the phrase "`the screen`" must be a position.

In summary, the idea of treating reference evaluation (RE) and source disambiguation (SD) as a CSP is based on the fact that the characteristics of these two processes are suitable to be described as a CSP. In the rest of this chapter, we will talk about how source disambiguation and reference evaluation can be formalised as an integrated CSP and how to find the solution to that CSP. During the discussion, (5.2) will be used several times to demonstrate how constraint satisfaction is used to resolve both source and referent ambiguities.

5.4 Formalising RE and SD as a CSP

In the last section, we argued that RE and SD could be viewed as an integrated CSP. In this section, we will talk about how to build such a CSP.

5.4.1 Binary CSP simplification

We assume that only unary and binary relations happen in our system. This means that all the constraints/preferences in the CSP are either unary or binary, so the CSP is a *binary CSP*.

The appropriateness of this simplification lies in:

- *It simplifies the modelling of the problem.* We do not need to consider any situation involving constraints/preferences that are not unary or binary relations.

- *It does not seriously limit the ability of the model.* Rossi and his colleagues point out that it is possible to convert CSP with n-ary constraints into an equivalent binary CSP if all the variables have a finite and discrete domain [Rossi et al., 1989]. As we will show in the following discussions, each variable in our CSP has a finite and discrete domain.

- *The majority of the restrictions on sources and referents are unary or binary relations.* When we talk about the origins of the constraints and preferences in Section §5.4.4 and §5.4.5, we will explain the types of constraints/preferences to be generated from each origin. The reader can notice that most of those constraints/preferences represent unary or binary relations, which is the justification of the above claim.

- *It enables us to use many existing resolution algorithms for binary CSPs.* For example, we use Mackworth's network consistency approach [Mackworth, 1977] to resolve the CSP, because it is simple and works at a local level. In addition, the graphic network representation of a binary CSP provides a clear presentation of the CSP.

5.4.2 Sources and referents: the variables of the CSP

Variable types

When a problem is formalised as a CSP, the variables are those components that would receive values after resolving the problem. Usually, the fewer the variables, the simpler the CSP. Therefore, it is a good strategy to represent two components by one variable if they are conceptually identical.

The first type of variable in our CSP represents sources. According to the above strategy, since the source of a word is relevant to the source of an entity whose attribute is represented by the word, the same variable can be used to represent the source of an entity, the source of the attribute of the entity and the source of the word raised from that attribute.

Another type of variable in our CSP represents entities. As the results of referent evaluation, the intended referents of phrases and probably some described entities would be found. At the beginning of the CSP, they are all represented as variables for future instantiation. For example, the intended referent of the referring expression "`the blue car`" can be represented as a variable $IR1$.

We mentioned the source of an attribute, but we did not say anything about the attribute itself. Should it also be represented as a variable?

The answer depends on the environment setting of the system. If lexical ambiguities are allowed, the attribute raised by the word would need to be a variable. However, as stated in Section §1.3.2, we are more interested in source ambiguities, and we assume that there is no lexical ambiguity in the input sentences. As a result, attributes are represented as constraints in the CSP (to be discussed in the next section).

We have mentioned that conceptually identical components would better be represented by the same variable in the construction of a CSP. However, this only suits the conceptual level. In the actual computation process, we may have to adjust our position a little bit. If we are not sure about whether two components are conceptually identical or not, two different variables would be used. A concrete example is the representation of the described entities of a referring expression.

According to the definitions in Section §3.3, described entities are derived from most parts of a referring expression (i.e., adjectives, nouns and prepositional phrases in our current linguistic coverage). Some of the words give rise to the same described entities, whereas others raise different ones. Before the described entities are actually known, it is hard to tell which words raise the same described entities, and which ones do not. Therefore, to simplify the computation, we define that each word in a referring expression that needs a source give rise to a variable that represents an individual described entity. The results of the CSP will tell which variables actually represent the same conceptually described entity. For example, variables {*DE11, DE12, DE21, IR1, IR2, Sde11, Sde12, Sde21, Sir1,Sir2*} are formalised from (5.2), and their meanings are:

- Variables `DE11, Sde11, DE12, Sde12, IR1` and `Sir1` are from the phrase "`the blue car`". Variables `DE21, Sde21, IR2` and `Sir2` are from the phrase "`the`

screen".

- The described entity variable DE11 represents the described entity related to the predicate "is_blue", which represents the word "blue" of "the blue car". The source of the entity is represented by source variable Sde11.
- The described entity variable DE12 represents the described entity related to the predicate "is_car" and its source, which represents the word "car" of "the blue car". The source of the entity is represented by source variable Sde12.
- The intended referent variable IR1 represents the intended referent of the phrase "the blue car". The source of the entity is represented by source variable Sir1.
- The described entity variable DE21 represents the described entity related to the predicate "is_screen", which represents the word "screen" of "the screen". The source of the entity is represented by source variable Sde21.
- The intended referent variable IR2 represents the intended referent of the phrase "the screen". The source of the entity is represented by source variable Sir2.

Variable domains

The other issue about variable formalisation for a CSP is to decide their domains. In some application areas, the domains of variable could be indefinite and hard to define. Fortunately, the domains of all the variables in our CSP are finite and discrete. This is because the source, no matter affiliating to an attribute or an entity, only takes one of the two values, the screen or the domain. Therefore, all variable representing a source have the same domain: {screen, domain}.

Theoretically, before being ruled out by a constraint, any entity in the databases can be the described entity or the intended referent. Therefore, the domains of variables representing described entities and intended referents are the same, that is, the set of all the entities in the databases. However, for computation purposes, we want the domains of variables to be as small as possible. So we need to consider different types of referring expressions individually.

In Section §3.4.1, we talked about classifying noun phrases according to their referent origins. We can also define the domains of variables representing described entities and intended referents for the types of phrases defined by their referent origins.

The described entities and the intended referents of definite noun phrases are among the entities in the context model and the display model and their corresponding entities (to cope with quasi-coreference). Therefore, the domain of a variable representing a described entity or an intended referent of a definite noun phrase is the set containing these entities. Similarly, the domain of a variable representing a described entity of an indefinite phrase is the set containing the entities in the world model and the display model. The domain of a variable representing a described entity or an intended referent of a deictic phrase is the set of the possible pointed entities in the display model and their corresponding entities. The domain of a variable representing a described entity or an intended referent of a pronoun is the set containing the entities in the context model and their corresponding entities.

Consequently, variables *Sde11*, *Sde12*, *Sde21*, *Sir1* and *Sir2* have the same domain {`screen`, `domain`}, and variables *DE11*, *DE12*, *DE21*, *IR1* and *IR2* have the domain containing all the entities in the context and display model and their corresponding entities.

Variable representation

Representing variables is a necessary step in the constraint satisfaction process, and it should provide the information the process needs.

Firstly, the information about the domains of variables is obviously needed in the constraint satisfaction process. Since all the variables have a finite and discrete domain, the two main concerns of the representation are:

1. how to enumerate possible values of variables and
2. how to reflect the number of candidates in the solution of a variable

When enumerating items, a naturally suitable data structure is *set*. Indeed, when Mellish considered the representation for the variables of the CSP of incremental pro-

cessing of referring expressions he used *set* [Mellish, 1985, p. 20]. The entities that could potentially be the results of a variable are enumerated in a set, which is called the *candidate set* of that variable.

The answer to the second question is straightforward in our situation. Because an entity can only have one source, either the domain or the screen, the number of candidates in the solution of a source variable is always 1. In the mean time, we only consider singular referring expressions whose intended referents can only be one entity. This means that our resolution model always tries to resolve the solution of a intended referent variable of a definite referring expression to be one entity regardless whether the entity is unique or the most salient one. In addition, our intention of using a variable to represent a described entity implies that its solution has only one candidate. On the whole, the number of entities in the solutions for all the variables in our representation is one, so this number will be treated as the default, and would not appear in the actual representation of variables.

The set representation with the default number 1 satisfies another requirement of the constraint satisfaction process. As we mentioned, variables can be in three different stages during the process (i.e., unresolved, partially resolved, and fully resolved). The set representation makes variables in different stages share the same structure. This avoids the transformation among different representations in different stages, and therefore saves time and space. In addition, by checking the number of elements in the set against the default value 1, the constraint satisfaction process knows which stage the variable is in. So it knows when the solution has been found so it should stop.

For example, suppose the representation of variable $IR1$, which represents the intended referent of the phrase "`the blue car`$_d$", has a candidate set {`icon1, icon2, icon3`}. If later on its solution is found to be **icon3**, the candidate set would become {`icon3`}.

Variable representation also has to consider information related to the variables representing entities. The CSP to be built will not only resolve source ambiguities and find referents for phrases that have referents, such as definite noun phrases, deictic phrases and pronouns, but also resolve the source ambiguities for phrases that do not have referents, such as indefinite phrases and interrogative phrases. Although, theo-

retically, a CSP can only succeed when all the variables have found solutions, we have a slightly different success criterion. We do require the CSP to find solutions to all the variables representing sources and the entities of phrases having referents, but we do not pose such obligation to the CSP to find a solution to every variable that represents an entity of a phrase not having a referent. For this reason, there is information in the representation of a variable to indicate whether the variable represents an entity of a phrase that has a referent or not.

In summary, the information needed in a variable representation is

- the domain of the variable, which is represented as a set;
- the information that the entity represented by the variable is from a phrase with or without a referent.

5.4.3 Restrictions: the constraints and preferences of the CSP

In a binary CSP, variables can be viewed as nodes, and restrictions as links between them. When combining them together, the constraint satisfaction system becomes a network. Just as not all the components of a system can be formalised as variables, not all the features of variables or relations between variables can be used as restrictions in a CSP. Only those that are locally computable/inferable and easily comprehensible are qualified [Rich and Knight, 1991].

In Section §5.2.3, we mentioned that constraints can be discovered differently. By considering the flexibility of the CSP system we built and the complexity of resolving source and referential ambiguities, we decided to have the CSP system generating constraints but merely in the formalisation stage. In the following sections, we will talk about how the constraints are generated in the formalisation process. Once they are generated, they will be sent to the constraint satisfaction process for resolving and no new constraint would be generated in there.

Constraints and preferences

In Section §3.11, we mentioned the principle of favouring those rules that we feel more confident to use. This gives us a mechanism for handling contradictory evidence. The principle has been integrated in the formalisation of the restrictions of our CSP. We call obligatory rejection relations *constraints*, whereas heuristic preference relations *preferences*. If a value does not satisfy a constraint, it is rejected, whereas a preference to a particular value is optional. We also assign different objectives to the constraint propagation process and the search process in the satisfaction process. The constraint propagation aims at getting rid of any value in a candidate set that does not satisfy a constraint, whereas the search process tries to select a particular value by using a preference in order to initiate a new constraint propagation. If a contradiction is found, all the applied preferences would be checked and one of them would be relaxed. The checking and relaxing process does not apply to constraints.

(5.3) "Is the blue icon to the right of the red one small?"

For example, in (5.3), the restriction that *the source of the word "blue" and that of the word "red" shall be the same* is just a preference (according to the same type rule), whereas the restriction that *the source of the word "blue" shall be the same as that of the word "icon"* is a constraint (according to the screen head noun rule).

However, there are indeed cases where contradictory evidence cannot be resolved by dividing evidence into constraints and preferences. It is not uncommon to have a contradiction from two constraints or two preferences, for example. Currently, we do not have a theoretically sound solution. The method we have implemented is to let the time at which a constraint or a preference is added into the constraint stack or the preference stack decide which one has more authority (see Section §6.4.2). In the future, it would be possible to further divide evidence based on more studies of source ambiguities.

5.4.4 Constraints/preferences on sources

As we addressed at the beginning of this chapter, our constraint satisfaction process tries to achieve two goals, source disambiguation and referent evaluation. This is reflected by the types of variables in the CSP, which represent sources and entities separately. The same applies to modelling constraints and preferences. Some constraints/preferences restrict the source of attributes or entities, whereas others restrict entities, including described entities and intended referents. The constraints/preferences can be clustered into two groups: *constraints/preferences on sources* and *constraints/preferences on entities* (including those on described entities and intended referents). As the reader will see, the contents of constraints/preferences within the same group are similar, whereas those in different groups are significantly different.

Constraints on sources state the criteria that any possible value of a source should satisfy, and *preferences on source* state that a particular source is preferred.

The origins of constraints/preferences on sources

Constraints on sources are restrictions derived from the following origins of knowledge:

1. *domain specific knowledge about the sources of some particular attributes.* For example, in the Car Selection domain, words "`screen`" and "`icon`" are always related to entities from the screen, which means their sources are always the screen. Similarly, words "`car`" and "`expensive`" always have the domain as their sources. Therefore, the source variables raised from these words would be given a constraint saying that "`the solution of variable` A `must be the source` a", which can be written as $must\_be(A, a)$. Constraints obtained this way are unary constraints.

2. *regularity rules, such as the screen head noun rule* (see Chapter 3 or Section §5.1). For example, there are three source variables generated from the phrase "`the blue icon`$_s$". They are variable $S1$ representing the source of the described entity related to the word "`blue`", variable $S2$ representing the source of the intended referent of the phrase and variable $S3$ representing the source of the

described entity related to the head noun "icon". According to the screen head noun rule, the values of $S1$ and $S2$ are the screen. The constraints related to this situation can be written as $must\_be(S1, screen)$ and $must\_be(S2, screen)$. Constraints obtained from this rule are unary constraints.

3. *the semantic preconditions of some relations or operations.* For example, if an attribute is possessed by an entity, the source of the attribute should be the same as that of the entity. Therefore, because of the possessive relation denoted by "of" in (5.4), the variable $S4$, which represents the source of the described entity of the phrase "the top", has the same value as the variable $S5$, which represents the source of the intended referent of the phrase "the screen_s". In addition, the variable $S6$, which represents the source of the move operation, has the same value as the variable $S7$, which represents the position mentioned by the phrase "the top of the screen".

   (5.4) "move the blue car to the top of the screen".

   These two constraints can be written as $same\_source(S4, S5)$ and $same\_source(S6, S7)$. They are binary constraints.

Preferences on sources are usually from the following origin of knowledge:

4. *the heuristic rules mentioned in Section §3.6, 3.7 and 3.10.* For example, the word "red" and the word "yellow" in (5.5) are preferred to come from the same source because they satisfy the same type heuristic rule. This means that the variables raised from these two words, say $S7$ and $S8$, are preferred to have the same value. This can be written as $prefer\_same(S7, S8)$. Preferences are binary.

   (5.5) "Delete the red car at the right of the yellow car.",

The constraints on sources for (5.2) are given in Table 5.1, where Con1 and Con2 are generated from the knowledge origin 1, Con3 from the origin 2, and Con4 from the origin 3.

Con1	$must\_be(Sde12, domain)$.	Con2	$must\_be(Sde21, screen)$.
Con3	$must\_be(Sir2, screen)$.	Con4	$same\_source(Sir1, Sir2)$.

Table 5.1: The constraints on sources raised from (5.2)

5.4.5 Constraints/preferences on entities

Constraints on entities can come from various origins, which are enumerated as follows. The first three were introduced by Mellish [Mellish, 1985], and the last two are extra restriction origins raised in our problem.

1. *Local semantic information that derives directly from the meaning of the components of a referring phrase.* In our CSP, the components are the adjectives, nouns, and prepositional phrases in a referring expression. For example, the noun "`car`" in the phrase "`the nissan`$_d$ `car`$_d$" provides a constraint on the variable representing the described entity, which says that the value of the variable must be a car.

   In (5.2), the constraints raised by local semantic information from the two referring expressions "`the blue car`" and "`the screen`" are shown in Table 5.2.

Con5	"$DE11$ `has blue feature, that is` $has\_feature(DE11, blue)$."
Con6	"$DE12$ `has car feature, that is` $has\_feature(DE12, car)$."
Con7	"$DE21$ `has screen feature, that is` $has\_feature(DE21, screen)$."

Table 5.2: The local semantic constraints on entities raised from (5.2)

2. *Global restrictions that derives from various positions of a text, whose range can be within a sentence, or as broad as the whole dialogue.* In our system, global restrictions are usually the "preconditions" of operations that make the operations "meaningful" or "well-formed". For instance, in order to execute the `remove` operation in (5.2), the intended referent of the phrase "`the blue car`" should be removable by user's request to the system, and the intended referent of the phrase "`the screen`" should be a position[2] . The constraints from this

[2] We acknowledge that there could be a substantial amount of work if the preconditions are to be computed automatically. Therefore, they are pre-installed in the knowledge bases by hand (see Chapter 6).

category for (5.2) are shown in Table 5.3.

Con8	"$IR1$ `must be removable, that is` $has\_feature(IR1, removable)$".
Con9	"$IR2$ `must be a position, that is` $has\_feature(IR2, position)$".

Table 5.3: More constraints on entities raised from (5.2)

3. *Information from the syntactic analysis.* People working on coreference and disjoint-reference have already pointed out some restrictions between the intended referents of two phrases. For instance, the intended referent of the phrase "`the blue car`" and that of the word "`it`" in the sentence "`Is the blue car near it?`" cannot be the same entity.

   We are more interested in problems related to restrictions from local semantic information, global information and the two information origins to be mentioned below. Therefore, the restrictions from syntactic analysis will not be discussed further in this thesis.

4. *Relations between two variables. One variable represents an entity and the other variable represents the source of the entity.* Our variable formalisation scheme requires a source variable to represent the source of an entity, and an entity variable to represent the entity. There are straightforward restrictions between the two variables, i.e., "`the relation between the entity variable and the source variable is that they represent an entity and the source of the entity`". For instance, the constraints generated from this category for (5.2) are shown in Table 5.4.

Con10	$source\_entity(Sde11, DE11)$.	Con11:	$source\_entity(Sde12, DE12)$.
Con12	$source\_entity(Sde21, DE21)$.	Con13:	$source\_entity(Sir1, IR1)$.
Con14	$source\_entity(Sir2, IR2)$.		

Table 5.4: More constraints on entities raised from (5.2)

5. *the restrictions among the described entities and the intended referents of the same phrase.* According to the discussion about the described entities and the intended referent in Chapter 3, there are restrictions between a described entity and the intended referent, or between two described entities. In general, the restric-

tions are: "the intended referent of a referring expression is either the same entity as or the corresponding entity of a described entity" and "a described entity of a referring expression is either the same entity as or the corresponding entity of another described entity of the same referring expression". This type of constraint helps the CSP to restrict the conceptually identical variables to have the same value in the results.

For instance, in (5.2), the constraints generated this way are shown in Table 5.5, where $same\_or\_corres(A, B)$ means that A and B are either the same entity or the corresponding entities of each other.

Con15	$same\_or\_corres(DE11, DE12)$.	Con16	$same\_or\_corres(DE11, IR1)$.
Con17	$same\_or\_corres(DE12, IR1)$.	Con18	$same\_or\_corres(DE21, IR2)$.

Table 5.5: More constraints on entities raised from (5.2)

There could be preferences on entities too. Similar to those on sources, preferences on entities can be overridden by the results of the variables they apply to.

One resource of preferences on entities is the relation between the salience of a dialogue entity and the likelihood of that entity to be the referent. Many previous researches (such as [Walker, 1997]) used a heuristic that the more salient a dialogue entity is, the more likely it is to be the referent. Due to the complexity of human dialogue, this heuristic rule is not always valid. It says that "if a choice has to be made among values of a variable, and every value corresponds to an entity in the context model, the value connecting with the entity that has the most salience is preferred."

The other resource for preferences on entities is the spatial relation between an entity and the pointed position. People usually assume that the nearer an entity is to the pointed position, the more likely it is to be the entity that the user wants to pick up. The preference based on the rule says that "if a choice has to be made among values of a variable for the described entities or the intended referent of a pointing phrase, the entity that is nearest to the pointed position is preferred".

Now we have presented all the considerations about formalising variables and constraints/preferences for the source disambiguation and referent evaluation process. Again, we use the fully established constraint network for (5.2) (shown in Figure 5.2) to give the reader a complete picture. In the figure, the nodes are the variables, and the links are the constraints. The list of all the constraints are given in Table 5.6.

Con1	$must\_be(Sde12, domain)$.	Con2	$must\_be(Sde21, screen)$.
Con3	$must\_be(Sir2, screen)$.	Con4	$same\_source(Sir1, Sir2)$.
Con5	$has\_feature(DE11, blue)$.	Con6	$has\_feature(DE12, car)$.
Con7	$has\_feature(DE21, screen)$.	Con8	$has\_feature(IR1, removable)$.
Con9	$has\_feature(IR2, position)$.	Con10	$source\_entity(Sde11, DE11)$.
Con11	$source\_entity(Sde12, DE12)$.	Con12	$source\_entity(Sde21, DE21)$.
Con13	$source\_entity(Sir1, IR1)$.	Con14	$source\_entity(Sir2, IR2)$.
Con15	$same\_or\_corres(DE11, DE12)$.	Con16	$same\_or\_corres(DE11, IR1)$.
Con17	$same\_or\_corres(DE12, IR1)$.	Con18	$same\_or\_corres(DE21, IR2)$.

Table 5.6: All the constraints raised from (5.2)

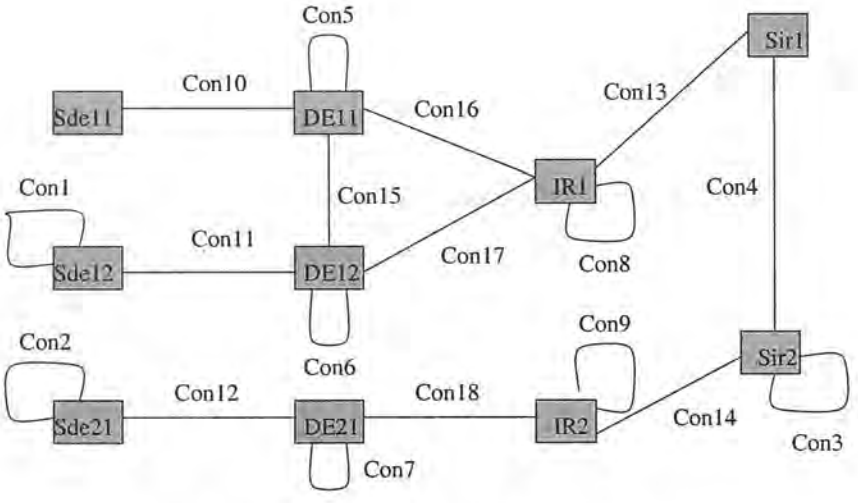


Figure 5.2: The constraint graph of (5.2)

5.5 Constraint satisfaction in the CSP

After a CSP is constructed, it can go through a process called **constraint satisfaction**, where the values in the candidate set of variables are checked against the constraints. Hopefully at the end of this process, the solutions of the variables can be found. In our research, CSP serves as a tool to resolve source ambiguities and referents. The

study of algorithms for constraint satisfaction is a side issue for us and we choose to use those existing methods that are appropriate for our problem.

5.5.1 Propagation

Usually, the constraint satisfaction process has two sub-processes. The first one is propagation. The method we chose to use in this sub-process is Mackworth's network consistency approach (see Section §5.2.4).

The version of network consistency approach we used here only achieves node and arc consistencies of a constraint network because it seems the most cost-effect way of finding solutions for our problems.

In the propagation process, only constraints of the network are used. As mentioned in Section §5.4.3, this arrangement is the result of our formalisation of constraints and preferences.

In order to help the reader to understand our constraint satisfaction process, we will use the example in Figure 5.2 during our discussion.

Since the two noun phrases in (5.2) are both definite noun phrases, the domains of the variables representing entities are the entities in the context model and the display model and their corresponding entities (see Section §5.3.2). Assume the entities in the two models are:

- the context model: {`car1, car2`}
- the display model: {`icon1, icon2, screen1`}

The attributes of these entities are stored in either the world model or the display model, and the mapping relations between these entities are in the mapping model. Suppose the contents of the models are:

1. the world model

```
car(car1), blue(car1), have_source(car1, domain),
car(car2), red(car2), have_source(car2, domain),
```

```
car(car3), white(car3), have_source(car3, domain)
```

2. the display model

```
icon(icon1), red(icon1), have_source(icon1, screen), removable(icon1),
icon(icon2), red(icon2), have_source(icon2, screen), removable(icon2),
screen(screen1), have_source(screen1, screen), position(screen1)
```

3. the mapping model

```
corres(car1, icon1), corres(car2, icon2)
```

The first three columns of Table 5.7 show the variables and their candidate sets in the initial state and after achieving the node consistency of the network.

variable	initial candidate set	after NC	after AC-3
Sde11	{screen, domain}	{screen, domain}	{domain}
Sde12	{screen, domain}	{domain}	{domain}
Sde21	{screen, domain}	{screen}	{screen}
Sir1	{screen, domain}	{screen, domain}	{screen}
Sir2	{screen, domain}	{screen}	{screen}
DE11	{*}	{car1}	{car1}
DE12	{*}	{car1, car2}	{car1}
DE21	{*}	{screen1}	{screen1}
IR1	{*}	{icon1, icon2}	{icon1}
IR2	{*}	{screen1}	{screen1}

Table 5.7: The candidate sets of the variables of the network in figure 5.2 where {*} = {car1, car2, icon1, icon2, screen1}

The last column contains the results after executing the following arc consistency process, which is just one of many sequences to achieve the consistency. In the following discussion, as a simplification for merely description reasons, when we say an arc $C_{x,y}$ is consistent, we mean both the arc from x to y and that from y to x are consistent. The arc consistency process can be described as:

1. Achieve arc consistency on the arc between $Sde12$ and $DE12$, and the candidate set of $DE12$ is still {car1, car2}, since both of the two candidates satisfy the constraint "have the domain source".

2. Achieve arc consistency on the arc between $DE12$ and $IR1$, and the candidate set of $IR1$ is still {icon1, icon2}.

3. Achieve arc consistency on the arc between $DE12$ and $DE11$, and the candidate set of $DE11$ and $DE12$ both become {car1}.

4. *Re-achieve* arc consistency on the arc between $DE12$ and $IR1$, and the candidate set of $IR1$ becomes {icon1}.

5. Achieve arc consistency on the arc between $DE11$ and $IR1$, and nothing changes.

6. Achieve arc consistency on the arc between $IR1$ and $Sir1$, and the candidate set of $Sir1$ becomes {screen}.

7. Achieve arc consistency on the arc between $DE11$ and $Sde11$, and the candidate set of $Sde11$ becomes {domain}.

8. Achieve arc consistency on the arc between $Sde21$ and $DE21$, and nothing changes.

9. Achieve arc consistency on the arc between $DE21$ and $IR2$, and nothing changes.

10. Achieve arc consistency on the arc between $IR2$ and $Sir2$, and nothing changes.

11. Achieve arc consistency on the arc between $Sir2$ and $Sir1$, and nothing changes.

12. Now the whole network is arc consistent.

variable	initial candidate set	candidate after NC	after AC-3
Sde11	{screen, domain}	{screen, domain}	{screen, domain}
Sde12	{screen, domain}	{domain}	{domain}
Sde21	{screen, domain}	{screen}	{screen}
Sir1	{screen, domain}	{screen, domain}	{screen}
Sir2	{screen, domain}	{screen}	{screen}
DE11	{*}	{car1, icon3}	{car1, icon3}
DE12	{*}	{car1,car2, car3}	{car1, car3}
DE21	{*}	{screen1}	{screen1}
IR1	{*}	{icon3, icon1, icon2}	{icon1, icon3}
IR2	{*}	{screen1}	{screen1}

Table 5.8: The candidate sets of the variables of the network in figure 5.2 where {*} = {car1, car3, icon3, icon1, icon2, car2, screen1} (on a different condition)

The above steps achieve the consistency of the network, and refine the domains of the variables. In the above example, when the network becomes consistent, each variable has only one candidate in its candidate set. This candidate is the solution. However,

solutions are not always appear when network consistency has been achieved. For example, suppose the problem is the same as the above one except that the entities in the context model are now {car1, car3, icon3} where **icon3** is the corresponding entity of **car3** and the details of **icon3** are

```
icon(icon3), blue(icon3), have_source(icon3, screen), removable(icon3),
```

As shown in Table 5.8, the network has not reach a solution even though network consistency has been achieved.

It is obvious that achieving network consistency does not equate to finding the solution. So, in many cases, the other process in constraint satisfaction is necessary, i.e., searching.

5.5.2 Searching

Searching starts when network consistency has been achieved throughout the network but the solution has not been found. From the variable perspective, this means that not all the variables needing solutions have the numbers of entities in their candidate sets reduced to one. Since all the constraints have been applied at this point, other resources are needed to resolve the problem.

Preferences formalised from the problem (i.e. those mentioned in Sections §5.4.4 and 5.4.5) are used in searching. Their function is to select from the corresponding candidates for a remaining variable whose solution has not been found. As mentioned, we do not have a theoretically sound solution for selecting preferences. The details of the method we have implemented are presented in Section §6.4.2. After a preference has been selected and applied, the propagation process can be resumed to achieve network consistency again. If the solution still cannot be found, another preference would be selected. For example, suppose we are interested in resolving sentence (5.6), and the results of some variables have not been found after achieving network consistency (see Figure 5.3 for the constraint network and Table 5.9 for the constraint details), then a preference based on the same type rule can be used to select the value of $Sde11$ to be {screen} because $Sde11$ and $Sde21$, which has value {screen}, satisfy the same type

rule. After applying the preference, the network needs to re-achieve its consistency and then the solution is found (see the last column of Table 5.10).

(5.6) "Delete the red car at the right of the yellow car.",

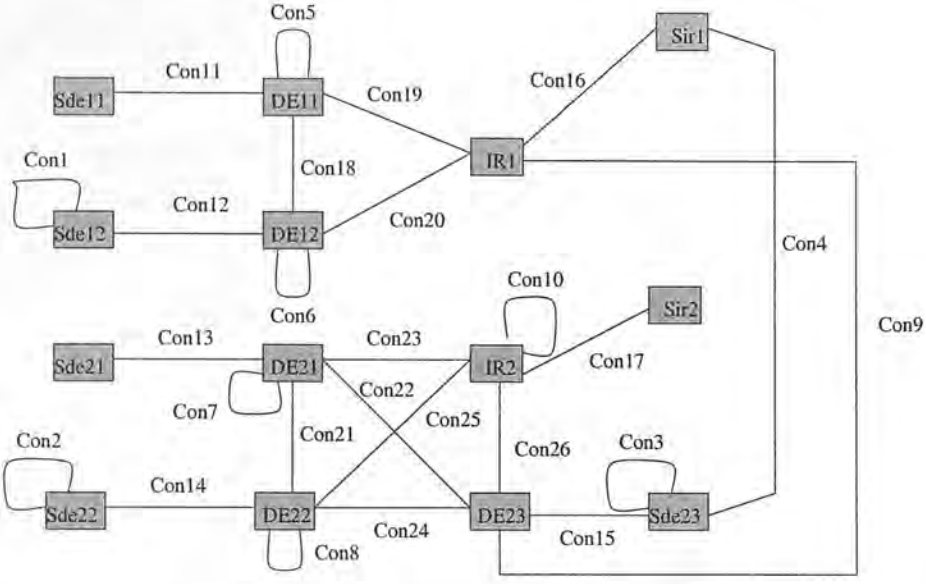


Figure 5.3: The constraint graph of (5.6)

Con1	$must\_be(Sde12, domain)$.	Con2	$must\_be(Sde21, domain)$.
Con3	$must\_be(Sde23, screen)$.	Con4	$same\_source(Sde23, Sir1)$.
Con5	$has\_feature(DE11, yellow)$.	Con6	$has\_feature(DE12, car)$.
Con7	$has\_feature(DE21, red)$.	Con8	$has\_feature(DE22, car)$.
Con9	$has\_feature(DE23, IR1, at\_right\_of)$.	Con10	$has\_feature(IR2, removable)$.
Con11	$source\_entity(Sde11, DE11)$.	Con12	$source\_entity(Sde12, DE12)$.
Con13	$source\_entity(Sde21, DE21)$.	Con14	$source\_entity(Sde22, DE22)$.
Con15	$source\_entity(Sde23, DE23)$.	Con16	$source\_entity(Sir1, IR1)$.
Con17	$source\_entity(Sir2, IR2)$.	Con18	$same\_or\_corres(DE11, DE12)$.
Con19	$same\_or\_corres(DE11, IR1)$.	Con20	$same\_or\_corres(DE12, IR1)$.
Con21	$same\_or\_corres(DE21, DE22)$.	Con22	$same\_or\_corres(DE21, DE23)$.
Con23	$same\_or\_corres(DE21, IR2)$.	Con24	$same\_or\_corres(DE22, DE23)$.
Con25	$same\_or\_corres(DE22, IR2)$.	Con26	$same\_or\_corres(DE23, IR2)$.

Table 5.9: All the constraints raised from (5.6)

During the process of achieving network consistency, entities in the context model are all treated the same. However, in the searching process, when considering salience, more salient entities are preferred under the assumption that the dialogue is coherent.

variable	initial candidate set	candidate after NC	after AC-3	after searching
Sde11	{screen, domain}	{screen, domain}	{screen, domain}	{screen}
Sde12	{screen, domain}	{domain}	{domain}	{domain}
Sde21	{screen, domain}	{screen}	{screen}	{screen}
Sde22	{screen, domain}	{domain}	{domain}	{domain}
Sde23	{screen, domain}	{screen, domain}	{screen, domain}	{screen}
Sir1	{screen, domain}	{screen, domain}	{screen, domain}	{screen}
Sir2	{screen, domain}	{screen}	{screen}	{screen}
DE11	{*}	{car1, icon3}	{car1, icon3}	{icon3}
DE12	{*}	{car1,car2, car3}	{car1, car3}	{car3}
DE21	{*}	{icon1, icon2}	{icon1, icon2}	{icon1}
DE22	{*}	{car1,car2, car3}	{car1, car2}	{car1}
DE23	{*}	{*}	{icon1, car2}	{icon1}
IR1	{*}	{*}	{car1, icon3}	{icon3}
IR2	{*}	{icon1,icon2,icon3}	{icon1, icon2}	{icon1}

Table 5.10: The candidate sets of the variables of the network of sentence 5.6 where {*} = {car1, car3, icon3, icon1, icon2, car2}.

For example, the preference based on salience information can be used for the unresolved variables in Table 5.8, which are shown in the first two columns of Table 5.11. According to the preference, **icon1** is selected as the solution to $IR1$ because its corresponding entity **car1** is more salient than **icon3**. Based on this decision, the solutions for all the variables are found after network consistency is achieved again (see the last column of Table 5.11).

variable	candidate sets after AC-3	candidate sets after searching
Sdel1	{screen, domain}	{domain}
DE11	{car1, icon3}	{car1}
DE12	{car1, car3}	{car1}
IR1	{icon1, icon3}	{icon1}

Table 5.11: The candidate sets of the remaining unsolved variables of the network in Figure 5.2 after achieving network consistency

The searching, which is based on preferences could sometimes fail. Since a failure is the result of an inconsistency, backtracking is used. Backtracking aims at getting rid of the inconsistency.

The inconsistency does not necessarily mean that the current constraint/preference is wrong. This just indicates that there is at least one contradiction among the constraints and preferences having been applied so far. To get rid of it, some of the applied constraints/preferences should be removed. Since constraints are compulsory relations that solutions must satisfy, only those applied preferences need to be reconsidered because they are heuristics which can be overridden. When all the preferences have been examined and the CSP is still not satisfied, an error message would be generated to inform the user.

One remaining issue that needs to be clarified about the constraint satisfaction process is when the process should stop. This is not necessarily as trivial as it may seem. Apparently, the process should stop when it has found the solutions for all the variables. This is true when the constraint network only has entity variables from phrases that need referents (i.e., definite noun phrases, deictic phrases and pronouns). However, if the network also tries to solve source ambiguities for phrases that do not need referents (i.e. indefinite noun phrases and quantified phrases), it could be difficult and, more importantly, unnecessary to find the solutions for variables from those phrases.

Therefore, the stopping criterion for our CSP is that all the source variables and the entity variables from the phrases that need referents have found their solutions. Please note that there is no such obligation for the entity variables from the phrases that do not need referents.

5.6 Summary

This chapter presents the idea of treating source disambiguation and referent evaluation as an integrated constraint satisfaction problem. We formalised variables and constraints from the original problems, and built a constraint satisfaction model for them. In this model:

- sources and entities that can be either described entities in the described entity set or the intended referents of a referring expression are formalised as variables. The domains of these variables are finite and discrete.
- Various restrictions and regularity rules for choosing the source values of words, described entities and intended referents are represented as constraints and preferences. A constraint is obligatory and it has to be satisfied by the values of its variables, whereas a preference can be overridden by the values of its variables.
- The task of finding the sources and referents is switched to the searching of satisfying assignments of variables under the restrictions of constraints and preferences.
- We choose a network consistent algorithm to find solutions to the CSP. During the process of achieving network consistency, node and arc consistency are used. Because network consistency is not the same as finding the solution to the problem, sometimes searching, and even backtracking, is needed for the final solution after network consistency has been achieved.

Chapter 6

The prototype IMIG system

In this chapter, we will talk about the prototype IMIG system in detail. The discussion, however, will concentrate on the implementation aspect of the system. The components of the IMIG system can be classified into three groups. The first group contains all the processes, i.e. the parser, the reference module, the scope binding module and the execution module. They form a processing pipe that transforms the input English sentences into MRL_S expressions for execution among the knowledge bases. The second group consists of the knowledge bases, i.e. the world model, the display model, the mapping model, the general model and the context model. The third group contains only the graphic interface of the IMIG system. It is implemented in Tcl/Tk, and allows user typing and pointing. A simple example is given at the end of this chapter to illustrate the functions of the various modules.

6.1 System overview

The IMIG system is an intelligent multimodal system involving two modalities, text and graphics. Its most notable characteristic is that it can process more complicated references in the input sentences, such as phrases with source ambiguities, than most intelligent multimodal systems.

Figure 6.1 shows the architecture of the IMIG system. According to the simplifications

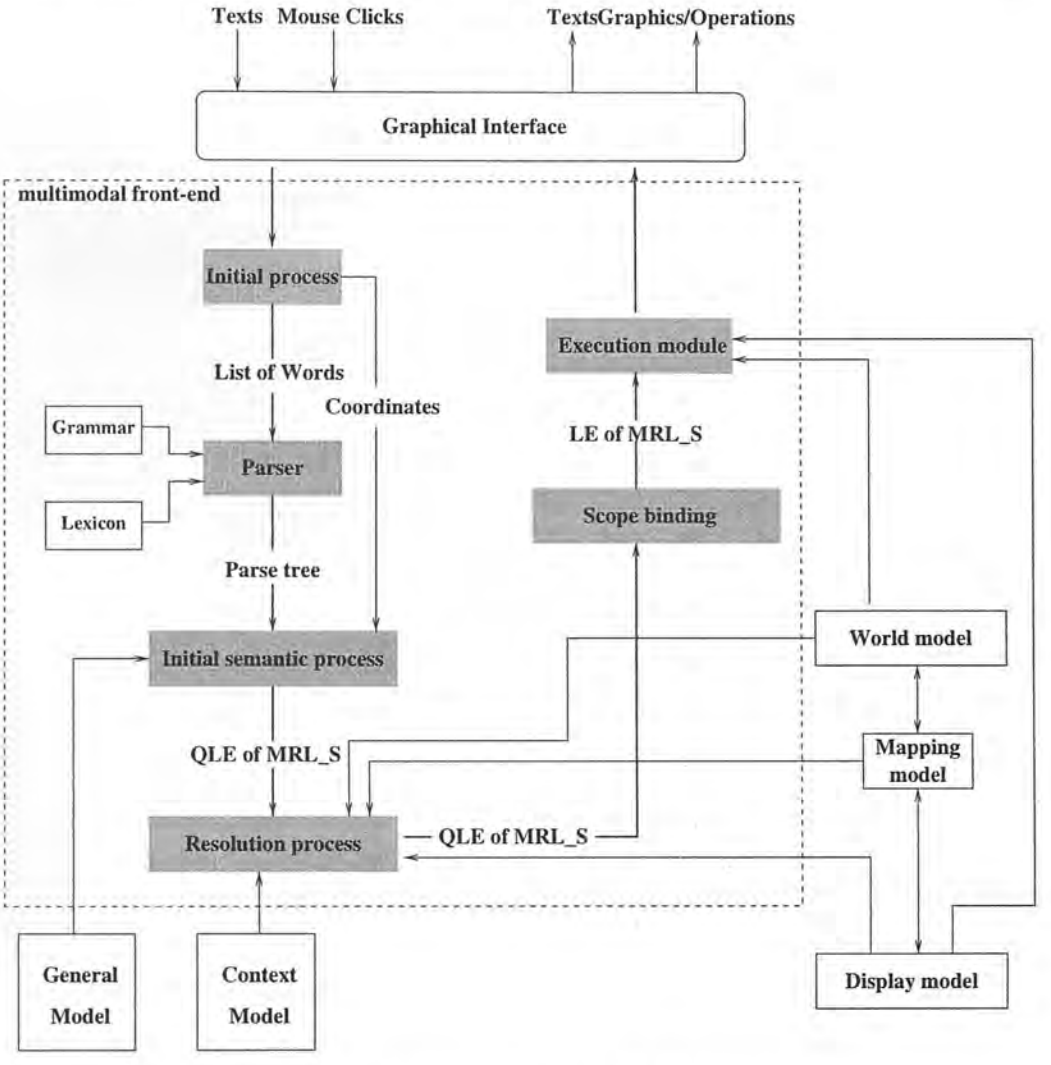


Figure 6.1: The structure of the IMIG system

given in Section §1.3.2, the allowed language form is written text, so English sentences are entered into the system through the keyboard. Besides, another input is the mouse clicks on the icons on the screen. Consequently, the IMIG system has three forms of input: text only, gesture only, and the combination of text and gesture.

The initial process module takes its data from the interface and transforms the string form text into a list of words for parsing. The parser analyses the list and generates the corresponding result, which could be multiple results if the sentence has syntactic ambiguities. The result(s) is(are) subsequently translated into a QLE expression(a set

of QLE expressions) in the initial semantic process module. If there is a pointing action (i.e. a mouse click) in the input, the results of the interpretation of the pointing action would be added into the QLE form during the translation. The benefit is that both linguistic and gesture information is available in the reference module for source and referent disambiguation. After both the source and referential ambiguities are resolved, the QLE will be sent to the scope binding module to resolve any scope ambiguity and eventually be translated into an LE expression, or LE expressions if there are scope ambiguities.

LE is the other level of MRL_S. It encodes the necessary information for the execution of the input in the execution module. If the execution succeeds, a response will be generated, whereas if it fails, an error message will be given. Whether it succeeds or fails, user friendly responses are generated to inform the user as much as possible so that the dialogue can keep on going. A response can be an English sentence or an operation on the graphics displayed on the screen, such as adding, moving and removing. If a response involves a referring expression, a sub-module located in the execution module would be used to generate appropriate referring expressions dynamically.

The IMIG system is a knowledge-rich system, and several databases are needed during the above process. It has a knowledge allocation strategy to organise its knowledge. For instance, the display model stores the information about icons on the screen, the world model contains the knowledge about entities in the application domain; and the mapping model maintains the corresponding relations between the elements or the features of elements in the world model and those in the display model. The necessary domain independent knowledge is stored in the general model.

In the following sections, we will talk about the implementation of the above modules and databases in detail.

6.2 The parser: From text to AVM

Some IMM systems aim at developing a parser that can deal with not only language input, but also input from other modalities. Therefore, the function of their parsers is set to be very powerful [Johnston, 1998]. In contrast, the parser of the IMIG system

was implemented to handle language input only. Following the common approach, we selected a subset of English grammar to contain all the phenomena needed by the system performance.

6.2.1 The grammar coverage

There is no corpus directly related to the situations modelled by the IMIG system, so an appropriate approach here seems to be to analyse the typical sentences/dialogues that we want the system to handle.

There are several simplifications in the linguistic aspect of the coverage, which were mentioned in previous chapters. They are restated below.

1. Every referring expression in the IMIG system has a singular concrete entity in the system as its intended referent. There is no referring expression whose referent is a set of entities or an abstract entity.
2. The post-modifier in a noun phrase is a prepositional phrase. Post-modifying clauses are not considered.
3. One-anaphora and ellipses are not considered.
4. All the input is yes-no questions, wh-questions and imperative commands.
5. There is no indirect speech.

Bearing these simplifications in mind, we will describe the linguistic coverage on several levels.

Word level

The open words in the IMIG system are nouns, adjectives and verbs. The function words are the pronoun "`it`", demonstrative words "`this`" and "`that`", adverbs "`here`", articles "`the`" and "`a`", and interrogative words "`which`" and "`what`".

Phrase level

From the grammatical aspect, a noun phrase in the IMIG system can be a definite noun phrase (e.g. "the small circle"), a deictic phrase (e.g. "this↖ car" or "here↖"), the pronoun "it", an indefinite noun phrase (e.g. "a table") or an interrogative phrase (e.g. "which car").

From the source aspect, a noun phrase can be a screen phrase (e.g. "a blue_s icon_s"), a domain phrase (e.g. "the expensive_d car_d"), a mixed-source phrase (e.g. "the small_s car_d"), and an unknown phrase (e.g. the pronoun "it")[1].

Sentence level

The sentence types that the IMIG system is intended to deal with were mentioned several times in previous chapters. They are yes-no questions, wh-questions, and commands. If taking into account the source aspect of a referring expression, the source of a verb or a predicative adjective in front of the referring expression, the typical sentences that shall be considered are[2]:

1. verb/adjective + phrase

   (a) domain verb/adjective + domain phrase

   "$\text{Place\_an\_order\_for}_d$ the expensive_d car_d."

   "Is the Nissan_d car_d expensive_d?"

   "Which Nissan_d car_d is expensive_d?"

   (b) domain verb/adjective + mixed-source phrase

   "$\text{Place\_an\_order\_for}_d$ the blue_s car_d."

   "Is the blue_s car_d expensive_d?"

   "Which small_s car_d is expensive_d?"

---

[1] Pronoun "it" is treated as an unknown phrase because its source aspect depends on its antecedent, which is not available until the antecedent can be found by some means.

[2] Each of the following sentences should be considered as a representative of a set of sentences which have similar features to those revealed by these representative sentences. The subscripts accompanying some words indicate the sources of those words in the context of these sentences. For example, the source of the word "blue" in these sentences are the screen, but it could also be the domain in some other contexts.

(c) screen verb/adjective + domain phrase

"Delete$_s$ the expensive$_d$ car$_d$."

(d) screen verb/adjective + screen phrase

"Delete$_s$ this↖ blue$_s$ icon$_s$."

(e) screen verb/adjective + mixed-source phrase

"Delete$_s$ the blue$_s$ car$_d$ at_the_right_side_of$_s$ the yellow$_s$ car$_d$."

2. verb/adjective + phrase + phrase

(a) domain verb/adjective + domain phrase + domain phrase

"Move$_d$ the expensive$_d$ car$_d$ to the garage$_d$"

(b) screen verb/adjective + domain phrase + screen phrase

"Move$_s$ the expensive$_d$ car$_d$ to the top$_s$ of the screen$_s$."

(c) screen verb/adjective + mixed-source phrase + screen phrase

""Move$_s$ the blue$_s$ car$_d$ to the top$_s$ of the screen$_s$."

(d) screen verb/adjective + screen phrase + screen phrase

"Move$_s$ the small$_s$ circle$_s$ to the top$_s$ of the screen$_s$."

Dialogue level

The main interesting points at the dialogue level are coreference and quasi-coreference, so typical dialogues are:

(6.1) **User:** Is this↖ car small?
System: `Yes, it is.`
User: Move it here↖.
System: `Yes, your command succeeds.`

(6.2) **User:** How much is the green car?
System: `14,200 pounds.`
User: It is too expensive. Remove it from the screen.
System: `The green car has been removed from the screen.`

More dialogues can be seen in Chapter 7 and Appendix B.

6.2.2 Choosing HPSG_QAS

It is difficult and time consuming to design a good parser from scratch. Therefore, we decided to select one of the available existing parsers and revise it to suit our needs. The candidates were MASQUE [Auxerre, 1986], ANLT [Grover et al., 1993], HPSG_QAS [Seldrup, 1995] and NLITDB [Androutsopoulos, 1996].

We chose HPSG_QAS for the following reasons:

- Its linguistic coverage is reasonably close to our need, which saves us time on modification and extension.
- Its grammar is based on Head driven Phrase Structure Grammar(HPSG) [Pollard and Sag, 1994]. Since HPSG seeks to model a language, in our case English, as a system of order-independent constraints on typed feature structures, any extension or modification of the grammar becomes much easier than some earlier approaches. This is exactly what we need as extension and modification are inevitable in adapting an existing grammar to another usage.
- Its grammar is coded in the ALE formalism [Carpenter and Penn, 1994], which is based on PROLOG. This is another advantage since most other components of the IMIG system are written in PROLOG. It is much easier to integrate several components when they are all coded in the same language.
- Its lexical entry editor provides a convenient tool for building our lexicon. The lexical definition for a word in HPSG is quite complicated, but words in the same category share a great amount of information in the definition. The lexical entry editor takes advantage of this characteristic and only asks the lexicon builder to provide the specific information for a particular lexical entry, which makes the task of building a lexicon much easier.

6.2.3 The modification

It is seldom the case that an existing system can be used directly for another purpose without any modification. In our adaptation of the HPSG_QAS parser, the major

changes are building a lexicon for the car selection domain and the kitchen arrangement domain, and extended the grammar to deal with commands and pointing phrases, such as a noun phrase with merely a determiner(e.g. "this↖").

The activities related to building the lexicon include adding appropriate nouns, verbs and adjectives. In addition, some prepositions are also needed in the lexicon because the input sentences may use spatial relations. Seldrup's lexical editor helps us to avoid the complicated definitions of words in HPSG. For example, the definition of an entry for the word "car" can be as simple as

(6.3)
```
make_noun car.
predicate is_car.
semantic_type car_type.
end_noun.
```

In HPSG, major linguistic objects are modelled by a feature structure called *sign*. A sign is usually described by an *attribute-value matrix* (AVM), and can represent a word, a phrase or a sentence. Therefore, any post-parsing sentence is in AVM form. For example, after the parsing, the sentence "is this↖ car small?" would become the AVM in (6.4).

(6.4)
```
phrase
QSTORE ne_set_quant
       ELT quant
           DET pointing
           RESTIND nom_obj
                   INDEX [0] car_type
                         GEN neut
                         NUM sing
                         PER third
                   RESTR ne_set_psoa
                         ELT psoa_arg
                             NUCLEUS ne_set_psoa
                                     ELT is_car
                                         INSTANCE [0]
                                     ELTS e_set
                         ELTS e_set
       ELTS e_set
SYNSEM synsem
       LOC loc
           CAT cat
```

```
            COMPS e_list
            HEAD verb
                 AUX plus
                 INV plus
                 MOD none
                 PRD boolean
                 VFORM fin
            MARKING marking
            SPR e_list
            SUBJ e_list
       CONT psoa_arg
            NUCLEUS ne_set_psoa
                    ELT is_small
                        INSTANCE [0]
                    ELTS e_set
NON_LOC non_loc
       INHERITED non_loc_1
                 SLASH e_set
       TO_BIND non_loc_1
               SLASH e_set
```

This AVM will then be processed in the initial semantic process module to obtain a QLE expression the further processing.

6.3 The initial semantic process module: from AVMs to QLE expressions

The main task of this process is to build a QLE expression for the input sentence from the AVM, or one of the AVMs if there are syntactic ambiguities.

Two kinds of information are needed in building a QLE expression. The first is the common linguistic information about each word and its relations, which can be obtained from the AVM. The second is about the source information for some attributes/operations mentioned in the AVM, which can be obtained from the system hierarchy.

An AVM like the one in (6.4) contains a great amount of data for the purpose of showing parsing results. However, information and the notations of the information in an AVM could be different to its corresponding QLE expression. There has to be a translation, which has two steps: firstly, the useful information for a QLE expression

is extracted from the AVM, and then the extra annotations necessary for building the QLE are added.

The extraction involves taking information from the QSTORE and SYNSEM parts of an AVM. Using the AVM in (6.4), the relevant information that shall be extracted is

- From QSTORE:
  - the determiner `pointing`,
  - the entity that is an object in `car_type`, with the information `GEN:neut, NUM:sing, PER:third`,
  - the predicates `is_car` whose argument is the object in `car_type`
- From SYNSEM:
  - the predicate `is_small` whose argument is also related to the object in `car_type`.

In general, the main operations in constructing a QLE expression are:

- Translate a predicate into `qpred` form.
- Translate all constants into `cons` form because their sources are assumed to be clear without ambiguities all the time (see Section §3.5).
- Translate noun phrases, including pronouns and interrogative phrases into `qterm` forms.

For instance, the QLE generated from the AVM in (6.4) is

(6.5)
```
[test, [qpred(is_small), qterm(pointing, qvar(x),
                [pred(is_car, domain), qvar(x)])
       ]
]
```

The translation of AVM to QLE has nothing to do about the intended referent or related information. For example, the part of AVM related to a definite article would

be translated directly to the corresponding QLE expression without considering where or what the intended referent is. Such information will be examined in the reference module.

6.4 The Reference module for resolving source and referential ambiguities

The reference module uses the CSP-based approach mentioned in Chapter 5 to resolve source and referential ambiguities. As illustrated in Figure 6.1, the reference module receives its input, which is a QLE expression, from the initial semantic process. For example, suppose the input sentence is "`remove the blue car from the screen`", then the QLE expression sent to the reference module is:

(6.6)
```
[act, [qpred(remove),
            qterm(the, qvar(x),
              [and, [qpred(is_blue), qvar(x)],
                    [qpred(is_car), qvar(x)]
              ]),
            qterm(the, qvar(y),
                [qpred(is_screen), qvar(y)])
          ]
]
```

The output of the reference module is still a QLE expression. In order to distinguish these two QLE expressions, the one from the initial semantic process is called the *input QLE*, whereas the one generated by the reference module is called the *output QLE*. The output QLE is different from the input QLE in that the sources of predicates/variables in it are filled in, and some `qterms` are substituted by the corresponding constants to represent the referent being found. For instance, suppose the intended referents of the phrases "`the blue car`" and "`the screen`" are the constants **car1** and **screen1** respectively. The output QLE expression of (6.6) is given in (6.7), where the term `qterm(the, qvar(x), ...)` is substituted by the constant `cons(icon1, screen)`, and `qterm(the, qvar(y), ...)` by `cons(screen1, screen)`.

(6.7)
```
[act, [pred(remove,screen), cons(car1,screen),
                            cons(screen1,screen)]
]
```

We have discussed how intended referents are computed in Chapter 5. In this section, we will address the issue of how to fit the CSP approach into the interpretation process of the user input, in particular, we will concentrate on how the variables and constraints are generated from the input QLE expression and the system databases, and how the solutions are integrated into the output QLE expression.

The constraint network is the basis of our CSP process, and it is built for the input QLE expression. In the following, we will talk about how to find the variables for the network first.

6.4.1 The variable formalisation from input QLE

For computational simplicity, we assume that the described entities raised by each word are different unless evidence to the contrary appears later (see Chapter 5). Because a word in a referring expression is represented in a QLE expression as a `qpred` inside a `qterm` whose quantifier is either "`the`" or "`this/that`", we designed a process to go through the QLE expression and generate a described entity variable for each such `qpred`. For instance, from the QLE in (6.6), the process would generate variables $DE11, DE12$ and $DE21$ for terms `qpred(is_blue)`, `qpred(is_car)` and `qpred(is_screen)` respectively. At the same time, because the sources of these described entities are also needed for the CSP, the process generates a corresponding variable to represent the source of each described entity, e.g. $Sde11, Sde12$ and $Sde21$.

The intended referent of a referring expression is another kind of variable in the CSP, and it is represented as a `qvar()` term in a `qterm` in a QLE expression. Similar to the treatment of described entities, we designed a process to generate variables for the intended referents and their sources. Generally, for each `qvar()` in a `qterm` whose quantifier is "`the`" or "`this/that`", new variables IR_i and Sir_i will be generated. For example, variables $IR1$, $Sir1$, and $IR2$, $Sir2$ are generated from the QLE expression in (6.6) for `qvar(X)` and `qvar(y)` respectively.

If the input sentence contains only definite noun phrases and deictic phrases, the variable abstracting process is complete after the above processes. However, there could be indefinite noun phrases and interrogative phrases in the input sentence. The

attributes mentioned in these phrases need their sources to be identified not just for capturing their meanings, but also for the processes in the execution module. In addition, the information derived from these attributes may help to restrict the values of the variables raised from the referring expressions. This is why we talked about how the CSP are constructed when there are indefinite noun phrases and interrogative phrases etc. in Section §5.4.2.

The variables abstracted from these phrases are the same as those from other noun phrases. That is, variables representing the intended referent and its source are generated for each phrase, that is a `qterm` whose quantifier is "`exists`", "`interrog`", or "`forall`". For each `qpred/pred`, variables representing a described entity and its source are also generated.

We want to re-emphasise here that the intention for including those variables is to find the sources of the attributes in these phrases (e.g. the source of the described entity variables), and not the intended referents as for other noun phrases that need intended referents. Therefore, it is all right if these variables cannot reduce their numbers of candidates to one (see Section §5.5.2).

6.4.2 The constraint discovering

Some constraints/preferences that are needed in the CSP can be found from the input QLE expression. However, the generation of more constraints/preferences needs the help from the "*origins of the constraints/preferences*" mentioned in Sections §5.4.4 and §5.4.5.

From the system performance perspective, it is not enough to merely know which kind of knowledge is needed for finding a constraint/preference. The system also has to know which database should be used to access the appropriate knowledge. In a sense, the following discussion addresses the requirement from the reference module to the databases. We will go through all the origins of constraints/preferences one by one, but the emphasis is on the issue of where to find them.

Finding constraints/preferences on sources

- The first origin of constraints on sources is the domain specific knowledge about a particular word. Two methods can be used to represent this kind of knowledge. The first is to treat it as a part of the lexical knowledge, and store it in the lexicon, where the source information that is necessary for finding the constraints on sources could be derived. For example, there could be a slot in a lexical item saying that the source of this word should be restricted to the `domain`.

  Using the knowledge in the hierarchy is another way to find the same constraints on sources. The hierarchy might be physically stored separately from the lexicon, as is the case in the IMIG system (see Section §6.7.2). This method requires the hierarchy to contain information about which database each concept belongs to. For example, as to be presented in Section §6.7.2, since the concept `cars` is subsumed by the concept `domain_entities`, its source should be restricted to the `domain`.

- Another origin of constraints/preferences on sources are the regularity rules and the two heuristic rules. They are domain independent knowledge, because they are derived from the linguistic analysis stated in Chapter 3. In the IMIG system, all domain independent knowledge is stored in a database called the *general model*.

- The third origin of constraints on sources is the semantic preconditions of relations and operations, which are domain dependent knowledge because relations/operations can have different interpretations in different domains. In the IMIG system, the world model is the place where knowledge about the current domain is stored, so the semantic preconditions are in the world model.

Finding constraints/preferences on entities

- The constraints on entities based on local semantic restrictions are derived from the meaning of a referring expression, which in turn is taken from the meanings of the words in the expression. Since the meanings are affected by lexical knowledge or conceptual knowledge, both the lexicon and the hierarchy can be the origins

of these constraints. In the IMIG system the lexicon or the hierarchy is required to provide this information.

- The constraints on entities based on global restrictions can be derived from almost anywhere in a text, which can be either the meaning of a particular word or a precondition of an operation. In the first case, the lexicon or the hierarchy is the possible resource, whereas in the second, the world model is the resource.
- **salience and nearness for preferences on entities**

In summary, the possible places in the IMIG system for determining constraints and preferences are the lexicon, hierarchy, general model and world model.

Organising constraints and preferences as stacks

Since each time only one constraint or preference can be applied during the resolution process, a data structure like a queue or a stack is needed to organise the constraints and the preferences respectively. We choose stack in the IMIG system. That is, the later a constraint or a preference is added into the corresponding stack, the earlier it is used in the resolution process. There is no particular reason for selecting stack except that the list data structure in Prolog provides a straightforward implementation of stack. There are two stacks in the IMIG system. One for constraints and the other for preferences. This is because they are used in different stages of resolution process. Since the earlier a constraint/preference is applied, the more influence it has on the results, our implementation actually gives priority to those constraints/preferences that are added later to the stacks when there are contradictions.

The discovering of constraints/preferences accompanies the formalisation of variables. When a variable is formed, all the constraints related to that variable are added to the stack. A QLE expression is read from left to right in order to form the variables. Again, this is just one way of implementing variable formalisation and constraint/preference discovering. We choose it for implementation considerations.

The rest of the CSP implementation is straightforward after the detailed algorithm and process for constraint satisfaction have been specified in Chapter 5. In the fol-

lowing section, we will not discuss how to find results from a constraint network, but concentrate on the issue of integrating the results of the CSP into the output QLE of the reference module.

6.4.3 The generation of output QLE

If the input sentence has only definite noun phrases or deictic phrases, the results of the CSP, as discussed in Chapter 5, would be the described entities, the intended referents and their sources. Both the described entities and the intended referents are in constant form.

In Section §6.4.1, we mentioned that each intended referent is explicitly expressed as a `qvar()` in the input QLE expression, whereas the described entities are implicitly mentioned through attributes `qpred()` in the same `qterm`. In the generation of the output QLE expression, the whole `qterm`, which represents the description of the phrase referring to the intended referent, is substituted by the corresponding intended referent constant.

Unlike the intended referent constants, the described entities obtained from the CSP do not appear in the output QLE expression. Firstly, the description of the phrase in the input QLE expression that gives rise to these described entities is substituted by the intended referent constant, so there is no place for these described entities in the output QLE expression. Secondly, described entities are used in our model only for finding the intended referent systematically. There is no need for them to be there when the intended referent has been found. In (6.6), two intended referent constants substitute the two `qterms`, and the result is given in (6.7).

However, if there are phrases like indefinite noun phrases and interrogative phrases in the input sentences the results would be different. The CSP will not try to find intended referents for these phrases, so there could be no intended referent constant being generated by the CSP. Therefore, the corresponding `qterm` would not be replaced in the output QLE expression. In this case, the sources of the attributes mentioned in these phrases, which are the same as those of the described entities of these phrases, are still required in the generation of the output QLE. Consequently, the results of

both the intended referent and the described entities are used in the generation of the output QLE expression. For example, the input QLE expression for the sentence "`is some blue car small?`" is:

(6.8)
```
[test, [qpred(is_small),
                qterm(some, qvar(x),
                      [and, [qpred(is_car), qvar(x)],
                            [qpred(is_blue), qvar(x)]
                      ]
                     )
       ]
]
```

Suppose after the processing, the sources of attributes `blue`, `car` and `small` are `domain`, then the output QLE expression is

(6.9)
```
[test, [pred(is_small, domain),
                qterm(some, var(x, domain),
                      [and, [pred(is_car, domain), var(x, domain)],
                            [pred(is_blue, domain), var(x, domain)]
                      ]
                     )
       ]
]
```

6.5 The scope binding module

Scope binding is a necessary step in language processing because scoping ambiguities often occur when the input sentence has more than one quantifier. The main function of the scope binding module in the IMIG system is to resolve any scoping ambiguity in the input sentence and generate the most plausible meaning. Also in this process, the QLE expression generated from the reference module is transformed into LE expression(s).

The algorithm used in this module is Hobbs and Shieber's scoping algorithm [Hobbs and Shieber, 1987]. It has sufficient coverage over the scoping phenomena in the IMIG system. Their representation and our representation of LE/QLE are essentially the same, especially for the representation of quantifier terms before and after the scoping. Before scoping, a quantifier term in Hobbs & Shieber's representation has a quantifier, a variable and a restriction, which is exactly the same as a quantifier term

in QLE. After scoping, a quantifier term in their algorithm has a body, in addition to the above three components, which is exactly the same as a quantifier term in LE. However, the task of carrying source information makes LE/QLE a different representation to Hobbs & Shieber's, so a transformation process is required, at least at the surface level. The transformation happens both before scoping, which translates the QLE expression into Hobbs & Shieber's format, and after scoping, which translates the post-scoped expressions into LE expressions.

Many quantifiers are deleted in the reference module due to the obtained intended referents for all the phrases that need one, but several quantifiers (e.g. existential, universal and interrogative quantifiers) could be left in the QLE expression when the expression entering the scope binding module. Multiple quantifiers could lead to more than one meaning being generated by the scoping algorithm, but only one of them represents what the user is trying to get across. Therefore, it would improve system efficiency if we could get rid of the irrelevant meanings.

One commonly adopted method is to set priorities among quantifiers [Alshawi, 1992]. The following two heuristics show the priporities among interrogative, universal and existential quantifiers work in most of our examples.

> **Scoping Rule 1**: If there is a universal quantifier and an existential quantifier in the QLE expression, the LE expression in which there is a wider scope for the universal quantifier is preferred to all other possible LE expressions.

Scoping Rule 1 is used in CLE [Alshawi, 1992] as well. The IMIG system covers only a small part of the general language usage described by the rule. An example is given in (6.10), where the LE expression in (a) is preferred to that in (b) for the input sentence "`is every blue car near a small car expensive?`"

(6.10) (a)
```
[test, quant(every, var(X, domain),
                [and, [pred(is_blue, screen),
                          [corresobj(domain), var(X, domain)]],
                      [pred(is_car, domain), var(X, domain)]
                ],
                quant(a, var(Y, screen),
                   [and, [pred(is_small, screen), var(Y, screen)],
                         [pred(is_car, domain),
```

```
                        [corresobj(screen), var(Y, screen)]],
                    [pred(is_near, screen),
                        [corresobj(domain), var(X, domain)],
                        var(Y, screen)
                    ]
                ],
            [pred(is_expensive, domain), var(X, domain)]))
    ]
(b) [test, quant(a, var(Y, screen),
            [and, [pred(is_small, screen), var(Y, screen)],
                [pred(is_car, domain),
                    [corresobj(screen), var(Y, screen)]],
            quant(every, var(X, domain),
                [and, [pred(is_blue, screen),
                        [corresobj(domain), var(X, domain)]],
                    [pred(is_car, domain), var(X, domain)]
                    [pred(is_near, screen),
                        [corresobj(domain), var(X, domain)],
                        var(Y, screen)]
                ],
                [pred(is_expensive, domain), var(X, domain)]))
    ]
```

Scoping Rule 2 is another commonly used heuristic in scope binding [Alshawi, 1992]. Its application is straightforward in the IMIG system because IMIG is a question/answering system and does not allow indirect speech.

> **Scoping Rule 2**: If there is an interrogative quantifier in the QLE expression, then the LE expression in which the interrogative quantifier is the outermost quantifier is preferred to all other possible LE expressions.

In (6.11), the LE expression in (a) is preferred to that in (b) for the input sentence "`which blue car near a small car is expensive?`"

(6.11)

```
(a) [list, quant(wh, var(X, domain),
            [and, [pred(is_blue, screen),
                    [corresobj(domain), var(X, domain)]],
                [pred(is_car, domain), var(X, domain)]
            ],
            quant(a, var(Y, screen),
                [and, [pred(is_small, screen), var(Y, screen)],
                    [pred(is_car, domain),
                        [corresobj(screen), var(Y, screen)]],
                    [pred(is_near, screen),
                        [corresobj(domain), var(X, domain)],
                        var(Y, screen)]
```

```
                ],
                [pred(is_expensive, domain), var(X, domain)]))
        ]
(b)  [list, quant(a, var(Y, screen),
                [and, [pred(is_small, screen), var(Y, screen)],
                      [pred(is_car, domain),
                         [corresobj(screen), var(Y, screen)]],
                quant(wh, var(X, domain),
                      [and, [pred(is_blue, screen),
                                [corresobj(domain), var(X, domain)]],
                            [pred(is_car, domain), var(X, domain)]
                            [pred(is_near, screen),
                               [corresobj(domain), var(X, domain)],
                               var(Y, screen)]
                      ],
                      [pred(is_expensive, domain), var(X, domain)]))
        ]
```

After determination of quantifier scopes, the LE expression will be generated. As said, the generation is straightforward. For example, (6.12) gives the LE expression of (6.9) after the scope binding.

(6.12)
```
[test, quant(a, var(x, domain),
                [and, [pred(is_car, domain), var(x, domain)],
                      [pred(is_blue, domain), var(x, domain)]
                ],
                [pred(is_small, domain), var(x, domain)]
            )
]
```

6.6 The execution module

The execution module is a major component of the *front end* of the IMIG system and it operates on the data from the scope binding module (see Figure 6.1). Its two main functions are handling the execution of input queries against the databases and generating appropriate responses according to the results of the execution.

The execution module has three components: *the control sub-module*, *the look up sub-module* and *the referring expression generation sub-module* (see Figure 6.2). The control and look up sub-modules cooperate to achieve the first function, and the control and referring expression generation sub-modules fulfil the second function.

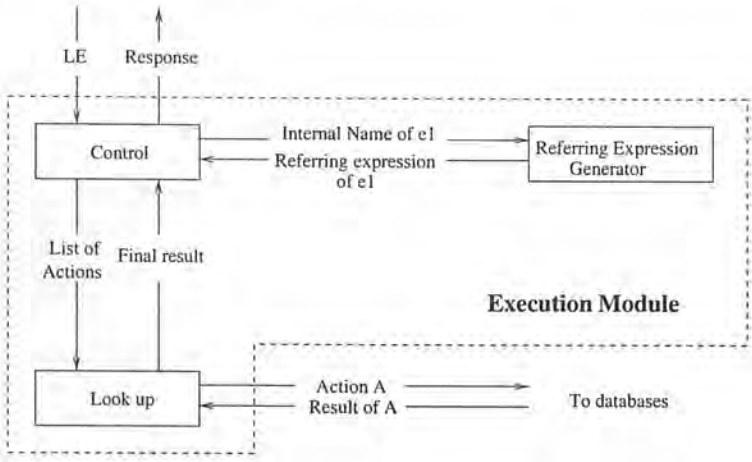


Figure 6.2: The structure of and the data-flow in the execution module

The input of the execution module is the LE expression generated by the scope binding module, and the output is a response based on the execution results.

6.6.1 The control sub-module

The control sub-module takes an LE expression, translates it into database operations, and then delivers these operations to the look up sub-module. It also generates appropriate responses based on the execution results sent back from the look up sub-module. The major processes in this sub-module are:

Translation

Translation is necessary because the LE of MRL_S is designed to represent the meaning of an input sentence. It might not be suitable for direct executions in the databases, especially if the databases are not in logical form.

Translation can have a general or a narrow interpretation in the control sub-module. The general interpretation concerns the portability of the system. The databases in the IMIG system can be represented in any database format. An intermediate translation between the LE format and the database format could minimise the impact of the change of database format on the definition of LE.

The narrow interpretation relates to the particular implementation of the current IMIG

system. There are several databases that may be accessed by the execution module, and they are constructed in PROLOG database format. The translation in this case is the decomposition of an LE expression into a sequence of PROLOG goals, which are used one by one in the look up sub-module for the execution.

Some transformations of the LE expression have to be done before it is translated into a list of PROLOG goals. The first is about universal quantifiers. Since our method of evaluating logical expressions in the execution is extensional (i.e., by enumerating and testing entities in the databases), it is usually much more complicated to evaluate a universal quantifier than that of an existential quantifier. For this reason, we translate a universal quantifier term into an existential term by using the transformation rule in (6.13). In the execution, if any entity is found to satisfy the expression $\neg F(A)$, the whole expression fails. Some other people evaluate their universal quantifiers in the same way [Seldrup, 1995].

(6.13) $$\forall x F(A) \Longleftrightarrow \neg\exists x \neg F(A)$$

Because LE expressions are all executed in the PROLOG environment in the look-up sub-module, a variable that is restricted by either a quantifier representing a wh-word or an existential quantifier would be instantiated to a particular entity regardless of the difference between the two types of quantifiers. In this sense, a wh-quantifier resembles an existential quantifier, and can therefore be substituted by an existential quantifier[3]. As a result, after the transformation of the universal quantifiers, the remaining quantifiers can all be seen as existential quantifiers, so they can be removed from the expression. In (6.14), the LE expression in (a) is transformed into (b) without changing its meaning.

(6.14) (a)
```
[test, quant(interrog, var(X, domain),
        [pred(is_car, domain), var(X, domain)],
          quant(a, var(Y, screen),
             [and, [pred(is_icon, screen), var(Y, screen)],
                   [pred(is_blue, screen), var(Y, screen)],
                   [pred(is_above, screen),
```

[3] This does not mean that the execution module would not care about the difference between these two types of quantifiers. Actually this difference is used in the control sub-module to generate appropriate responses for different types of input sentences. See the **Identification** part of this section.

```
                        [corresobj(domain), var(X, domain)],
                        var(Y, screen)]],
                    [pred(is_expensive, domain), var(x, domain)]))]
(b) [test, [pred(is_car, domain), var(X, domain)],
           [pred(is_icon, screen), var(Y, screen)],
           [pred(is_blue, screen), var(Y, screen)],
           [pred(is_above, screen),
                 [corresobj(domain), var(X, domain)],
                 var(Y, screen)],
           [pred(is_expensive, domain), var(x, domain)]]
```

A predicate term in LE has locally interpretable information about the predicate, its arguments, and their sources (see (6.14 b)), and its translation into a PROLOG goal is straightforward. A PROLOG goal has the form

```
call_action(LE_predicate, argument_list, source)
```

where `LE_predicate` is the predicate being translated, and `argument_list` is a list of arguments of the predicate. There is only one `source` in the goal because the predicate and its arguments always have the same source.

The arguments of a predicate can be either a variable (see (6.14)) or a constant (see (6.7)). When an argument is a variable, the special symbol `corresobj` for mapping relations can be used to represent the corresponding entity of the variable (see (6.14)). The PROLOG goal representation also encodes a function called `corresobj` to cope with the above situation. The list of PROLOG goals are embraced by the PROLOG predicate `exec_seq` to achieve the PROLOG variable convention. The translation result of (6.14) is shown in (6.15).

(6.15)
```
exec_seq(call_action(is_car, [X], domain),
         call_action(is_icon, [Y], screen),
         call_action(is_blue, [Y], screen),
         call_action(is_above, [corresobj(X, domain), Y],
                     screen),
         call_action(is_expensive, [X], domain)
        ).
```

Identification

Another function of the control sub-module is to deal with the result of the execution, that is, to interpret the success/failure of the execution of a yes-no question, a wh-question and a command.

The successful execution of a yes-no question means that the proposition being tested has a truth value `true` according to the current state of the databases. If the execution fails, the truth-value of the proposition is `false`.

In the case of a command, successful execution means that the goal of the command has been achieved. Otherwise, the execution of the goal fails.

From the perspective of the execution, a wh-question is similar to a yes-no question except that we are more interested in the entity being instantiated to the variable during the execution. As mentioned, during execution, the look up sub-module does not differentiate a wh-quantifier in a wh-question and an existential quantifier in a yes-no question, whereas the control sub-module, which knows the sentence type, can handle the task of discrimination. All the variables and their instantiated entities are returned to the control sub-module. There, the instantiated entities for wh-quantifiers are picked up and built in the responses shown to the user.

The control sub-module identifies the sentence type from the relevant information in the LE expression.

Generating responses

As an interaction system, the IMIG system has to generate appropriate responses to user queries so that the dialogue can continue. Generating a response is not as simple as just saying "`yes`" or "`no`" even for a yes-no question.

It is not user friendly to answer a yes-no question by merely saying "`yes`" or "`no`", although it is logically correct. An answer "`no`" can be understood as the falsity of the proposition mentioned in the question, or the failure of the execution itself. Therefore, more information has to be provided in the response. In the IMIG system, we use the following strategy to answer a `no` result of a yes-no question: if the proposition is false,

a message about the correct attribute of the object in the proposition will be given; if the execution fails, a message about this failure will be given. For example, if the blue car mentioned in the question "`is the blue car small?`" is actually a medium size car, the answer would be "`No, it is a medium size car.`", indicating that the proposition is false.

The above cooperative response requires the system to access the correct attribute through the execution module, which will be addressed further in the discussion of the look up sub-module.

Sometimes even a "`yes`" answer is not enough for a yes-no question. For instance, a user might raise a query like "`is there a small blue car?`" to indicate that s/he wants to search for such an entity. Recognising this intention, the IMIG system should provide an instance in its answer.

One interesting issue in introducing such an entity in the system's answer is that a referring expression is needed in order to bring the entity into the dialogue. This expression has to be generated dynamically when it is needed (This will be discussed in the introduction to the referring expression generator in Section §6.6.3).

Dynamic generation of a referring expression is also important in the answer to a wh-question, where an entity is found and should be presented to the user. For example, if the query is "`which car is small?`", and the entity in the answer is **`car1`** with a unique attribute `blue`, the referring expression "`the blue car`" may be generated.

When IMIG system cannot find an entity in the databases satisfying the requirements given by a wh-question, a pre-stated answer "`Sorry, the entity cannot be found.`" will be given to indicate the failure.

Compared with the treatment of the above two types of sentences, generating an appropriate response to a command is straightforward. Pre-stated sentences are used to tell the user whether his/her command has been executed successfully or not, e.g., "`Your command succeeds`" or "`Your command fails`". In addition, appropriate referring expressions would be generated for the entities in the command to keep the dialogue going. For example, a response to a move command could be "`Your command succeeds, and the blue car has been moved.`"

6.6.2 The look up sub-module

The look up sub-module reads the list of PROLOG goals from the control sub-module, and executes them against the databases. The process in this module is closely related to the actual representation of the databases, so any change to the database format could affect this sub-module.

One interesting issue here is how to achieve a PROLOG goal in which the predicate describes a relative relation, such as "`is_small`". The difficulty lies in that an entity may be qualified to be small in one situation, but not in another.

It is not correct to pre-compute and store the relative relations among entities because it is based on an invalid assumption, at least in the IMIG system, that the environment would not change before these relations are used. We choose to compute the relations dynamically when they are needed.

We assume that when a relation (such as "`is_small`") is about to be computed, the head noun of the phrase is known. The system knows from the hierarchy that the relation is about the size of an entity. So, it retrieves and orders the sizes of all the entities sharing the same category denoted by the head noun, say the sizes of all the cars if the head noun is "`car`". This gives a list of sizes, which indicates the available range of sizes for these entities. The range is then divided into three equal sub-ranges, which can be interpreted as satisfying the relation "`is_small`", "`is_medium`" or "`is_large`" respectively. According to the subrange the size of the tested entity is in, the above relation can be checked to be true or false for that entity. In this way, the relation "`is_smallest`" and "`is_largest`" can also be examined based on whether the size of the entity is the first or the last in the list.

Dynamic computation can also be used for interpreting spatial relations, such as "`is_south`", "`is_above`", etc. Similar to relative relations, spatial relations are based on the situations when the computation happens, and could change in different situations.

The computation of spatial relations is based on a data structure called a *generalised frame*. Each icon on the screen has a generalised frame that marks its territory. The

domain entities in a spatial domain like the kitchen arrangement domain also have a generalised frames, whereas those in a non-spatial domain, like the car selection domain, do not because there is no spatial relation between them. We will come back to this in Section §6.7.3.

Although the computations of relative and spatial relations were not mentioned until now, they could be used in the operations of some other modules, such as the CSP resolution process in the reference module. For example, for the phrase "`the small car near the red car`", the computations of both relative relation ("`near`") and spatial relation ("`small`") are needed. To cope with the demands from different modules and still keep modules independent, the computation part is kept in a common area where both the look up sub-module and the reference module can access.

Another interesting issue in implementing the look up sub-module is the interpretation of the special symbol `corresobj`. `corresobj` needs the mapping relations in the mapping model for its interpretation.

6.6.3 The referring expression generator

The referring expression generator is used to introduce a new entity into the dialogue when the user's query is a wh-question or a yes-no question with an indefinite noun phrase. (6.16) and (6.17) are examples of the two situations.

(6.16) **User:** Which car is expensive?
System: `The small blue car.`

(6.17) **User:** Is there an expensive car?
System: `Yes, the red ACMD car is an example.`

The referring expressions produced by the generator should be able to uniquely identify the referent in all situations. The generation is dynamic and should take into account the environment surrounding the entity.

Our referring expression generator is designed to be just powerful enough for the situations that might happen in our system. It is impractical and unnecessary to build a powerful generator that can deal with any situation. When asked to generate a description for a referent, we want our generator to be able to:

- generate a definite description, such as "`the blue car`", "`the small icon`", etc.
- generate a screen phrase, a domain phrase or a mixed-source phrase according to the situation.
- uniquely discriminate the referent from potential confusers[4].
- use all the attributes of the referent, including dynamically computed relative and spatial relations, alone or in combination with other attributes to compose a description.

We intend to rule out the following functions:

- *generate descriptions involving another entity, such as* "`the blue car near the red car`". Such phrases are difficult to generate and are unnecessary in the current system setting because the combination of the attributes of one entity is usually adequate.
- *check comprehensively about the realisation possibilities when selecting the content of a description.* It is still an open issue in Natural Language Generation (NLG) whether realisation should be considered during content selection. In our generator, several heuristics are used to make sure that the selected content can be realised as a grammatical description.
- *generate multimodal descriptions, such as* "`this`↖ `car`" *with a pointing action on the screen.* The difficulty lies in synchronising the appearance of a description in written form and a pointing action. It may be more practical to use such descriptions in spoken form.
- *generate pronouns.* The reason is that using a pronoun is not appropriate in the two situations that the generator is used.

The algorithm behind the generator is based on the work of Dale and Haddock [Dale and Haddock, 1991]. They state that the entity to be described has a set of attributes, and the task of the generator is to find a subset of the attributes that can

[4] A term borrowed from [Dale and Haddock, 1991]. See the next page for its definition.

distinguish the entity from all the other entities in the context. These other entities are called **potential confusers** to the entity to be described. The generator picks up an attribute from the set each time, checks the attribute against the remaining potential confusers and removes any confuser that does not have this attribute. If the attribute reduces the number of remaining potential confusers it is selected. Otherwise, another attribute is tried in the same way. This process keeps on going until all the potential confusers are ruled out, and then the attributes selected so far are used to generate the description. Obviously, this algorithm assumes that a grammatical description can be generated from the selected attributes. There could be cases where not all the potential confusers are ruled out even after all the attributes have been tried. This means that the attempt to generate a referring expression fails.

For example, suppose a unique reference is needed for the entity **car1**, whose attributes are {`car, blue, Nissan, small, cheap`}. The potential confusers are {`motorcycle1, car2, car3, car4`} and their attributes are those in (6.18). According to the above algorithm, the generator could select the attributes {`car, blue, Nissan`} and generate a referring expression "`the blue Nissan car`".

(6.18)
- motorcycle1: {`motorcycle, red, honda, large, expensive`}
- car2: {`car, blue, ford, small, cheap`}
- car3: {`car, black, benz, large, expensive`}
- car4: {`car, red, Nissan, medium, medium`}

In most cases, different referring expressions can be generated for an entity in the same situation. For example, phrases "`the small Nissan car`", "`the cheap Nissan car`" and "`the small blue Nissan car`" are all valid phrases for the above situation. Our generator just picks up the first one being generated.

6.7 Knowledge bases

6.7.1 Knowledge arrangement in the IMIG system

The IMIG system holds various sorts of knowledge. For easy maintenance and portability, we use a scheme to organise this knowledge without claiming that it is the best approach. The scheme is as follows:

- all knowledge about entities on the screen is stored in the display model,
- all knowledge about the particular application domain is in the world model,
- all knowledge about the mapping relations between entities/attributes in the display model and those in the world model is stored in the mapping model,
- general knowledge independent of any particular domain is stored in the general model,
- context information about previous dialogues is in the context model.

6.7.2 The hierarchy

As briefly explained in Section §3.6, the resolution model needs a knowledge hierarchy, which organises the concepts in the hierarchy into the following three types:

Type 1: concepts for entities and attributes/relations/operations that only appear in the world model, such as `cars, prices`.

Type 2: concepts for entities and attributes/relations/operations that only appear in the display model, such as `icons, delete`.

Type 3: concepts for entities and attributes/relations/operations that appear in both models, such as `colours, size`.

The benefit of this arrangement is that the hierarchical information can help in identifying the source for entities and features. In fact, the origin 1 of constraints/preferences on sources mentioned in Section §5.4.4 uses the knowledge within the hierarchy.

The IMIG system has implemented the knowledge hierarchy, and physically stores the concepts of Type 1, Type 2 and Type 3 in the world model, the display model and the general model, respectively. Therefore, if an entity or an attribute/relation/operation is a sub-category of a concept in the world model or the display model, its source is the domain or the screen respectively. If it is a sub-category to a concept in the general model, its source is unclear and we need more information to identify its source.

The topmost concept, which acts as the root of the hierarchy, is called `things`. Other concepts are organised as sub-trees going from abstract ones through less abstract nodes to more concrete leaves. In some parts of the tree, concepts that belong only to the display model or the world model appear. For example, in our current hierarchy, concepts at the roots of the sub-trees that are physically stored in the world model are `domain_entities`, `domain_quality_attributes`, `domain_quantity_attributes`, `domain_relations` and `domain_operations`. The concepts subsumed them are in the general model. Similar things happen in the display model.

6.7.3 The display model

The function of the display model

A display model is a common component of human computer interaction systems involving graphic display [Neal and Shapiro, 1991]. Its typical task is to provide necessary information for visual display. The display model in the IMIG system has the same function.

However, supporting the visual display is not the only function of our display model in the IMIG system. It is not even the most important function. The most important function of the display model is to provide higher level propositions that are used in language processing and database searching. In some other systems [Neal et al., 1988], only the world model can provide such function. The reason that this function becomes essential for our display model is because the IMIG system has to handle queries with source ambiguities.

The high level propositions related to the display model can be summarised into two groups: propositions related to entities and their attributes/relations/operations, and propositions concentrated on the preconditions of the operations mentioned in the input. Examples of the first group are whether an entity has an attribute or not, whether a screen entity is removable or movable, and so on. An instance of the second group is that removing operations can only be applied to a screen entity in the car selection domain. The knowledge representation in the display model has to be able to cope with both groups.

The content of the display model

To support the visual display, the display model provides the information about what to display on the screen. The information is sent to a component called the *rendering system* to visualise its content. According to the specification from a particular rendering system, the display model stores the appropriate details of the screen entities. For example, when a complex icon has to be decomposed into simple geometric objects for display, the display model would store the decomposition information explicitly.

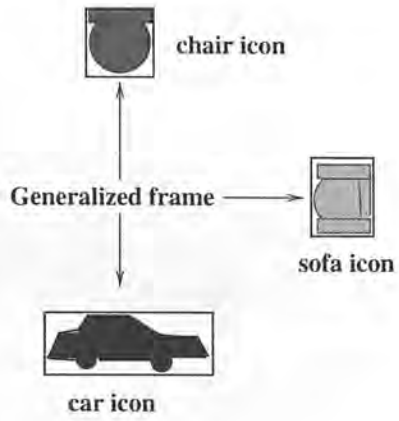
In addition, as mentioned in Section §6.7.2, related abstract concepts subsuming them in the hierarchy are also stored in the display model for possible retrieval during the processing.

A method widely used is to represent screen entities in an object oriented way [Bennett et al., 1996, Lee and Wang, 1996, Tang et al., 1996, Worboys, 1994]. The method is appropriate for the IMIG system because the screen entities in the system are all discrete entities. As an object, each screen entity has a unique identifier to distinguish it from other objects, and the object also contains some other essential attributes, such as its position on the screen, colour, texture, shape and size if they are available. Some attributes may come from inheritance.

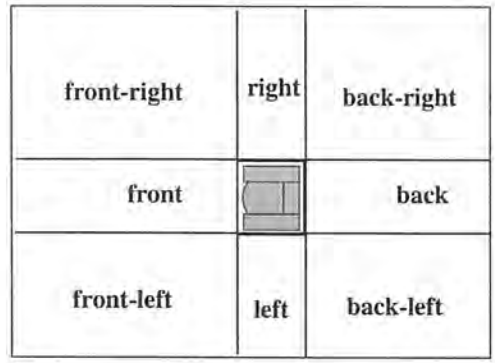
Because it is arguable that the whole screen display can be seen as a complex graph, it is interesting to ask at what point explicit representation of the information on the screen should stop. Should the spatial relations and relative relations be explicitly represented in the display model? As discussed in the execution module, our strategy is not to pre-compute and represent them, but to compute them at the spot when they are needed. The reasons are:

1. It is not possible to store all spatial relations and relative attributes of a screen entity when the number of entities is big.

2. It is inappropriate to assume that all the stored spatial relations and relative attributes would still be correct and appropriate for use when the display on the screen actually changes frequently during the interaction.



(a)



(b)

Figure 6.3: (a) the generalised frame of icons. (b) Eight parts of space and their name around an object

As mentioned before, each screen entity also has a data structure called **the generalised frame**, which is the smallest rectangular frame that can contain the icon entirely (see Figure 6.3).

The generalised frame can simplify the computation of a pointed entity. Pointing to an icon in the IMIG system requires that the mouse is moved into the area occupied by the icon and the left mouse button clicked. The central issue here is to identify whether or not the position pointed at is in the area occupied by the icon. Using a rectangular generalised frame, the identification is straightforward.

For simplicity and easy access, the preconditions of operations on the screen are treated as conjunction of individual precondition. For example the preconditions of operation `remove` can be written as the predicate in (6.19) where `Y` is the agent of the operation and `X` is the object of the operation.

(6.19) `precondition(remove(Y, X), [is_icon(X), removable(Y, X)])`

6.7.4 The world model

From the mapping relation perspective, the world model is the counterpart of the display model. It stores information about the current application domain, which is

domain dependent. Whenever the application domain is changed, the content of the world model has to be changed accordingly.

The function of the world model

The main function of the world model is to store information about domain entities and support any retrieval that is relevant to them. In particular, it is visited in the following situations:

1. when the system wants to display some domain entities on the screen. The command for displaying is usually accompanied by information about which entities and attributes of the entities are needed.

2. when the reference module tries to find the constraints on entities (see Chapter 3). In this case, the required information is the semantic preconditions of the operations in the world model.

3. when the reference module tries to resolve ambiguities. In this case, the required information is about whether or not a domain entity has some particular attributes/relations.

4. when the execution module tries to perform the user's queries. The required information in this case is similar to that in ambiguity resolution.

The content of the world model

The content of the world model is assumed to be unchangeable to the end user (see Section §1.3.2), so the functions for the world model are all retrieval processes. This is different from the case of the display model.

The world model contains the concept hierarchy that is domain dependent and has the domain source. All domain entities belong to a category in the hierarchy. Similar to screen entities, domain entities are also represented as objects with feature slots and appropriate values. For example, a blue Nissan car, with engine size 1.4 litre, price £4500, and M registration number, is represented in the world model as

(6.20)
```
domain_object(
    internal_name: car1,
    colour: blue,
    manufacturer: Nissan,
    engine_size:1.4,
    price: 4500,
    reg_number: M
    ).
```

The world model also contains the preconditions of domain operations. Similar to the preconditions of screen operations, the preconditions of domain operations are implemented as a list. For example the preconditions of operation buy can be written as `precondition(buy(Y, X), [has_price(X)])`.

6.7.5 The mapping model

The functions of the mapping model

One of the main characteristics of the IMIG system is that it has a mapping model, which contains explicit information about the connections between domain entities/attributes and screen icons/attributes. The mapping model is essential in the process of connecting domain entities/attributes with the corresponding screen icons/attributes. In addition, the mapping relations can also be used by other components of the system in the language understanding process.

Mapping relations are one-to-one representation relations between domain entities/attributes and screen icons/attributes. Because usually only a few attributes can be used in graphic display (e.g. *size, texture, colour, orientation* and *shape* [Bertin, 1983]), whereas many more domain attributes might be required to be displayed on the screen, it is very likely that mapping relations are always the assignments between a subset of domain attributes and the screen attributes. It depends on the situation of each specific interaction to decide exactly which subset of domain attributes are to be displayed on the screen and which domain attribute is mapped to which screen attribute. To achieve some degrees of flexibility, for each domain attribute in a given domain, we have a predefined list of screen attributes that are suitable for representing it. Therefore, the list can provide alternatives when a wanted screen attribute

has been occupied by other domain attributes.

The content of the mapping model

Required by the functions mentioned above, the mapping model has the following essential components:

An object pair list (OP list). This is useful when information about the relation between domain entities and screen entities is needed in the language understanding process, e.g. in the interpretation of function `corresobj`. It can also be used to give information about which domain entities are on the screen and which screen entities represent them. Since screen entities can be added and deleted during the interaction, this list is changable.

A mapping relation set (MR set). This set stores the mapping relations between domain attributes and screen attributes. There are two groups of mapping relations in the set. An *attribute mapping relation* is about the relation between a kind of domain attribute and a kind of screen attribute, that is, it states the relationship between two types of attributes. A *value mapping relation*, on the other hand, is about the relation between a domain value and a screen value when the attribute mapping relation has been decided. For example, the mapping relation between the domain attribute **the price of a car** and the visual display attribute **colour** is an attribute mapping relation, whereas the representation relation between domain value **price in 2,000 - 2,500** and screen attribute **blue** is a value mapping relation. The MR set is automatically generated after the user decides which domain attributes should be displayed.

A package of lists for simple mapping relation generation. Theoretically, any screen attribute can be used to represent a domain attribute, but there are usually restrictions in terms of convention, comprehensibility and visibility. Connecting domain attributes with screen attributes can be seen as defining the semantic interpretations of an image. Different semantic interpretations (i.e. different mapping relations) have different *effectiveness*[5]. The aim of mapping relation

[5] This term is borrowed from [Mackinlay, 1987].

generation is to achieve as high effectiveness as possible (the best combination of comprehension, visibility and fitting in convention, for example). This is itself an interesting research topic. However, what we have done in the IMIG system is very simple. We select and order screen attributes based on our perceptual thinking of their effectiveness in a particular representation. This is written as a list of ordered screen attributes for a domain attribute, and the list is used in the generation of mapping relations. An example of such list is "*the preferred graphic attributes for domain prices, with decreasing priority, are the size, the brightness of a colour, the change of texture and the caption*". Obviously, the ordering of the lists related to the domain attributes has effect on the generated mapping relations. In current version of IMIG, the ordering was decided randomly during the implementation.

Default display features. Certain information is essential for the display of an object, e.g. its size, colour, texture, orientation, shape and position. However, not all of them can be obtained from the mapping relations all the time. Therefore, a default value for each essential information is stored in the mapping model to be used when the mapping relations do not provide a value of a screen feature. If the default value has been used by other icons, an error message would be generated to indicate the failure of automatic mapping. Default values are decided during the construction of the domain application by the system organiser.

6.7.6 The general model

The general model is a knowledge base for storing domain independent knowledge. It is also the top part of the hierarchy that subsumes the parts in the world model and the display model.

The top of the hierarchy is a concept called `things`, which has several children, such as `entities`, `attributes`, `relations` and `operations`. Among the offspring of these concepts, some are connected with a concept in the display model, whereas others are connected with a concept in the world model.

In Section §6.4.2, we mentioned that the constraint discovery process needs the regu-

larity rules about sources. These rules are domain independent knowledge, so they are stored in the general model. The representation of regularity rules is straightforward. For example, (6.21) shows our representation of the part of the screen head noun rule (RULE 3.4) which is about the relation between the source of a screen head noun and that of the modifiers. Each rule is represented in the form `rule([], [])`, where the first `[]` contains a list of conditions of using the rule. The conditions are in conjunction. The second `[]` contains the result of using the rule.

(6.21)
```
rule([is_headnoun(X), source_of(S1, X), must_be(S1, screen),
      is_modifier(Y), source_of(S2, Y), same_phrase(X, Y)],
     [same_source(S1, S2)])
```

6.7.7 The context model

The context model is a knowledge base storing the entities in the context of the dialogue. In the IMIG system, the entities in the context can be either domain or screen entities.

The context model supports the resolution of anaphoric phrases in the input sentences by providing necessary information to the retrieval operations. In addition, the context model is revised to record the latest changes in the context after a turn of conversation (e.g. the user queries something, and the system gives an answer).

When updating the context model, the added entities can be *old* or *new*. An *old* entity is already in the model, so the update only changes its position in the model. However, a *new* entity is not in the context model before the update. Before talking about the situations that would introduce new entities into the context model, we first restate one of our arguments raised in the review of previous research in Chapter 2.

Previous research in intelligent multimodal interfaces tends to assume that all the entities on the screen are in the context model (or the dialogue model)
[Neal and Shapiro, 1991, Huls et al., 1995], but we think differently.

The context of a dialogue simulates an area in the short-term memory of the brain [Walker, 1997] where information about the salient entities mentioned in the dialogue is stored. This information is instantaneously accessible and easily forgotten. This

is why entities in the context would typically be removed after a while. However, a screen entity is usually assumed to be remembered until it is removed from the screen. For this reason, we argue that a screen entity is quite different from an entity in the dialogue context. In Chapter 2, we mentioned that we treat entities in the display model the same as those in the world model. The screen entities will be added into the context model only after they are mentioned in the dialogue (through either linguistic means, gesture or both), and they will be removed when they ought to be forgotten.

As to when to introduce a new entity into the context model, the following cases are relevant. In fact almost the cases related to the context model have been mentioned in previous discussion except the issue of introducing new entity into the context model. The following is a summary or a list of conditions implemented in the IMIG system for bringing new entities into the context model.

In the IMIG system, a *new* entity can be mentioned and thus should be added into the context model when:

- there is a pointing phrase, such as "`this`↖ `blue car`". In this situation, the screen entity being pointed is introduced into the context.
- there is a definite noun phrase in the visual situation use, such as "`the blue car`" where there is no such an entity in the context model, but there is one on the screen. In this situation, the screen entity is introduced into the context.
- there is a co-operative answer to a wh-question, e.g. the system's answer in (6.22 a) . Based on the answer to the question, either a screen entity or a domain entity can be introduced into the context.

(6.22) (a) **User:** Which car is expensive?
System: `The blue car.`

(b) **User:** Is some blue car expensive?
System: `Yes, such as the Nissan car.`

- there is a proper name, such as `car1`, although it is very unlikely that the user would uses a proper name directly because s/he could seldom know the name of

an entity. However, if the user does uses one, either a screen entity or a domain entity can be introduced.

- there is an indefinite noun phrase in a question, and the IMIG system tries to introduce a particular entity in its cooperative answer (see the system's reply in (6.22 b)). Based on the source of the entity, either a screen entity or a domain entity can be introduced.

- there is a pure screen operation like displaying some entities on the screen or moving an icon from one place to another. In this case, the screen entity is introduced into the context.

  This situation is special because it does not get any input from the language part at all. In the current IMIG system, pure screen operations include displaying, moving and removing. Entities manipulated by displaying and moving operations should be treated the same as those in input sentences because these entities may be referred to in the following sentences. For example, suppose a blue car is moved by the user through pure screen operation, then it can be the topic of the immediate following question, say "`is it cheap?`". So, the operated entities should be added to the context when the operation is adding or moving.

  The entities manipulated by a removing operation are not put into the context model because it is natural to think that a removed entity would usually not appear in subsequent dialogues. However, we acknowledge that this assumption is based on the fact that we do not handle sentences with past tense.

The structure of the context model

Adopting a previous approach on modelling the context of dialogue [Walker, 1997], the context in the IMIG system is organised as a fixed-size cache. To simulate the forgetting mechanism, the earlier an entity enters the model, the earlier it is removed from the model unless it is mentioned again. In the light of previous work [Kintsch, 1988, Alshawi, 1987], the context model contains the context information for the last three turns of conversion. Anything beyond the last three turns will be forgotten.

The intended referents of referring expressions in each turn are stored in the context

model. These entities are organised according to the *dialogue salience* where entities in one turn are more salient than those in previous turns, and inside a turn, the intended referent of the subject phrase is more salient than that of the direct object which is then more salient than that of the indirect object and other phrases. More salient referents are at higher positions in the model. For example, (6.23) is a snapshot of the context model, where `[[car1, domain], [icon2, screen]]` is from the current turn, `[[car2, domain]]` is from the last turn and `[car1, domain]` is from the turn before last turn. Here, `[car1, domain]`, `[car2, domain]` and `[icon2, screen]` are all the intended referents of the referring expressions in the turns.

(6.23)
```
[
  [[car1,domain], [icon2, screen]],
  [[car2, domain]],
  [[car1, domain]]
]
```

6.8 The user interface

6.8.1 Requirements on the interface

The IMIG system integrates natural language, graphics and pointing gesture modalities, so its interface supports natural language input/output, graphical output and pointing inputs. To be more precise, the requirements to the interface are:

1. The user can enter his/her queries in the interface, and the queries can be echoed on the screen.

2. The system's responses together with the user's queries should be displayed on the screen in a dialogue style.

3. There are icons displayed on the screen, so a graphical display area should be provided for showing these icons. This graphical area should be separated from the text area to get a clear display layout.

4. Mapping relations that are used by the screen icons should be listed on the screen because without them the screen display would be meaningless.

5. The interface should provide facilities to enable the user to use some screen operations in the conversation.

6. The interface should provide some common functions that exist in most interfaces, e.g. help information, and exit button, etc.

6.8.2 The Tcl/Tk based interface

The interface is implemented in Tcl/Tk. Tcl is an embeddable script language developed by John Ousterhout, and Tk is a GUI toolkit and widgets based on Tcl [Ousterhout, 1990, Ousterhout, 1991]. They are small but powerful. Many graphical interfaces have been developed based on their use [USENIX, 1998].

Choosing Tcl/Tk has another practical advantage. The core of the IMIG system, including the parser, the reference module and lots of other modules and databases, is implemented in Sicstus PROLOG [SICStus, 1996]. Accompanying the PROLOG package, there is a well defined application programming interface (API) to Tcl/Tk. So a Tcl/Tk based interface can be integrated easily with the rest of the IMIG system. The Tcl/Tk based interface is shown in Figure 6.4. It satisfies all the requirements mentioned in Section §6.8.1.

The control zone

The interface is divided into five zones. The top zone is called *the control zone.* It contains a number of push-down buttons. The `domain` button lists the application domains that the system can deal with. At the moment, there are only two choices: the car selection domain and the room arrangement domain.

The `dialogue` button pops up two choices: *to start a new dialogue* or *to terminate an existing one.* By selecting the start, the user can tell the system to empty its context for a different dialogue. At any time, only one dialogue is allowed. A new dialogue can be started only when the application domain has just been selected and no dialogue about this domain has been performed, or the existing dialogue has just been terminated.

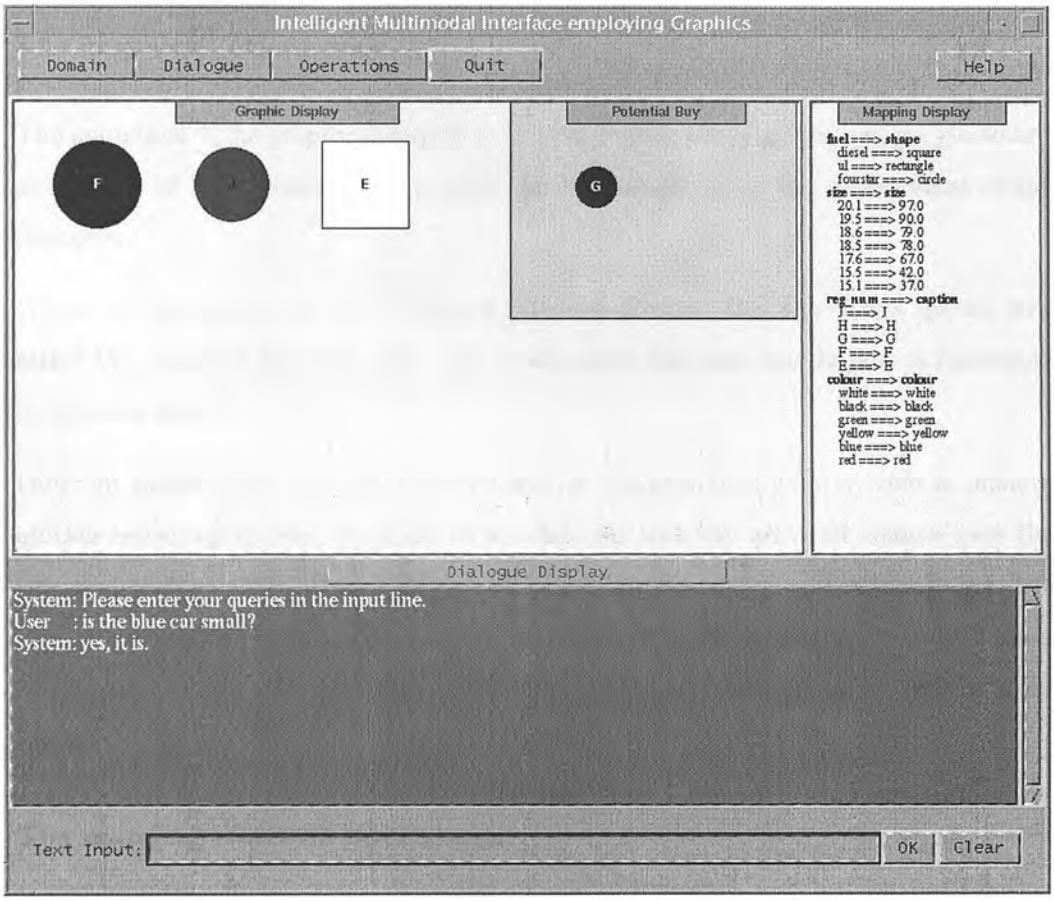


Figure 6.4: The Tcl/Tk based graphical interface of the IMIG system

Otherwise, an error message is given.

The `operations` menu provides the three pure screen operations that the user can use. They are explicitly shown in the interface to help inexperienced users to use them.

The `quit` and `help` buttons are straightforward. When they are pressed, the system will terminate its interaction with the user or provide help information.

The graphical display zone

The main task of the graphical display zone is to display the graphical images generated as a result of user requests. This zone can be thought of as the visual focus of the dialogue.

When the application domain is the car selection domain, this zone has a special area called *the potential buy area*. The user could move the cars that he/she is interested in into this area.

Different mouse clicks are also implemented in the graphical display zone to achieve quicker screen operation. To point to an icon, the user can move the mouse over the icon and click the left button. To move an icon, he/she can move the mouse over the icon, press and hold the middle button, and then drag the icon to the intended area. To remove an icon, the user can move the mouse over the icon and click the right button.

The mapping relations display zone

The mapping relations appearing in this zone are those used in displaying the screen entities, which include both the attribute mapping relations and the value mapping relations.

The dialogue display zone

This zone shows the user's input and the system's responses. By displaying these texts, it provides a linguistic context for the dialogue.

The text input zone

The text input zone is the place where the user types in his/her queries. The `OK` button signals the completion of the user's current query so that the system can process it and echo the user's input in the dialogue display zone. The `Clear` button cancels the text in the input line.

6.9 A simple example

To give a clear picture of the processes discussed in the previous sections, we provide a simple example.

Suppose the input sentence is like (6.24 a), and the coordinates of the pointing position is `(103,256)`. (6.24 b) is the list of words generated by the initial process model and it is also the input to the parser.

(6.24) (a) "`Which car near this↖ blue car is cheap?`"
(b) {`which, car, near, this, blue, car, is, cheap`}

(6.25 a) is the AVM generated by the parser, based on which a QLE (6.25 b) is generated.

(6.25) (a)
```
phrase
QSTORE ne_set_quant
       ELT quant
           DET pointing
           RESTIND nom_obj
                   INDEX [0] car_type
                         GEN neut
                         NUM sing
                         PER third
                   RESTR ne_set_psoa
                         ELT psoa_arg
                             NUCLEUS ne_set_psoa
                                     ELT is_blue
                                         INSTANCE [0]
                                     ELTS e_set
                         ELTS ne_set_psoa
                              ELT psoa_arg
                                  NUCLEUS ne_set_psoa
                                          ELT is_car
                                              INSTANCE [0]
```

```
                                                    ELTS e_set
                                          ELTS e_set
                ELTS ne_set_quant
                     ELT quant
                         DET interrog
                         RESTIND nom_obj
                                 INDEX [1] car_type
                                       GEN neut
                                       NUM sing
                                       PER third
                                 RESTR ne_set_psoa
                                       ELT psoa_arg
                                           NUCLEUS ne_set_psoa
                                                   ELT is_near
                                                       ARG1 [1]
                                                       ARG2 [0]
                                                   ELTS e_set
                                       ELTS ne_set_psoa
                                            ELT psoa_arg
                                                NUCLEUS ne_set_psoa
                                                        ELT is_car
                                                            INSTANCE [1]
                                                        ELTS e_set
                                            ELTS e_set
            ELTS e_set
SYNSEM synsem
       LOC loc
           CAT cat
               COMPS e_list
               HEAD verb
                    AUX plus
                    INV minus
                    MOD none
                    PRD boolean
                    VFORM fin
               MARKING marking
               SPR e_list
               SUBJ e_list
           CONT psoa_arg
                NUCLEUS ne_set_psoa
                        ELT is_cheap
                            INSTANCE [1]
                        ELTS e_set
       NON_LOC non_loc
               INHERITED non_loc_1
                         SLASH e_set
               TO_BIND non_loc_1
                       SLASH e_set
```

(b)
```
[list, [qpred(is_cheap),
       qterm(interrog, qvar(X),
            [and, [qpred(is_car), qvar(X)],
                  [qpred(is_near), qvar(X),
                   qterm(pointing, qvar(Y),
                         [and, [qpred(is_car), qvar(Y)],
```

```
                                    [qpred(is_blue), qvar(Y)]
                                ]
                               )
                        ]
                ]
                )
            ]
        ]
```

In the reference module, in order to construct a constraint network for the input, the variables for the described entities, the intended referents and their sources are abstracted from the QLE. They are {*DE11, DE12, DE21, DE22, IR1, IR2, Sde11, Sde12, Sde21, Sde22, Sir1, Sir2*} whose meanings are:

- Variables `DE11, Sde11, DE12, Sde12, IR1` and `Sir1` are from the phrase "`this↖ blue car`". Variables `DE21, Sde21, DE22, Sde22, IR2` and `Sir2` are from the phrase "`which car near this↖ blue car`".
- The described entity variable `DE11` and the corresponding source variable `Sde11` are from the predicate "`is_blue`".
- The described entity variable `DE12` and the corresponding source variable `Sde12` are from the predicate "`is_car`".
- The intended referent variable `IR1` and the corresponding source variable `Sir1` are from the phrase "`this↖ blue car`".
- The described entity variable `DE21` and the corresponding source variable `Sde21` are from the predicate "`is_car`".
- The described entity variable `DE22` and the corresponding source variable `Sde22` are from the predicate "`is_near`".
- The intended referent variable `IR2` and the corresponding source variable `Sir2` are from the phrase "`which car near this↖ blue car`".

The constraints for these variables are also generated (see Table 6.1), so the constraint network is constructed (see Figure 6.5).

Con1	$must\_be(Sde12, domain)$.	Con2	$must\_be(Sde21, domain)$.
Con3	$same\_source(Sde22, Sir1)$.	Con4	$has\_feature(DE111, blue)$.
Con5	$has\_feature(DE12, car)$.	Con6	$has\_feature(DE21, car)$.
Con7	$has\_feature(DE22, position)$.	Con8	$has\_feature(IR1, position))$.
Con9	$has\_feature(IR2, price)$.	Con10	$has\_relation(IR1, DE22, near)$.
Con11	$source\_entity(Sde11, DE11)$.	Con12	$source\_entity(Sde12, DE12)$.
Con13	$source\_entity(Sde21, DE21)$.	Con14	$source\_entity(Sde22, DE22)$.
Con15	$source\_entity(Sir1, IR1)$.	Con16	$source\_entity(Sir2, IR2)$.
Con17	$same\_or\_corres(DE11, DE12)$.	Con18	$same\_or\_corres(DE11, IR1)$.
Con19	$same\_or\_corres(DE12, IR1)$.	Con20	$same\_or\_corres(DE21, DE22)$.
Con21	$same\_or\_corres(DE21, IR2)$.	Con22	$same\_or\_corres(DE22, IR2)$.
Con23	$must\_be(Sir1, screen)$.	Con24	$must\_be(Sir2, domain)$.

Table 6.1: All the constraints formalised from the QLE in (6.25 b)

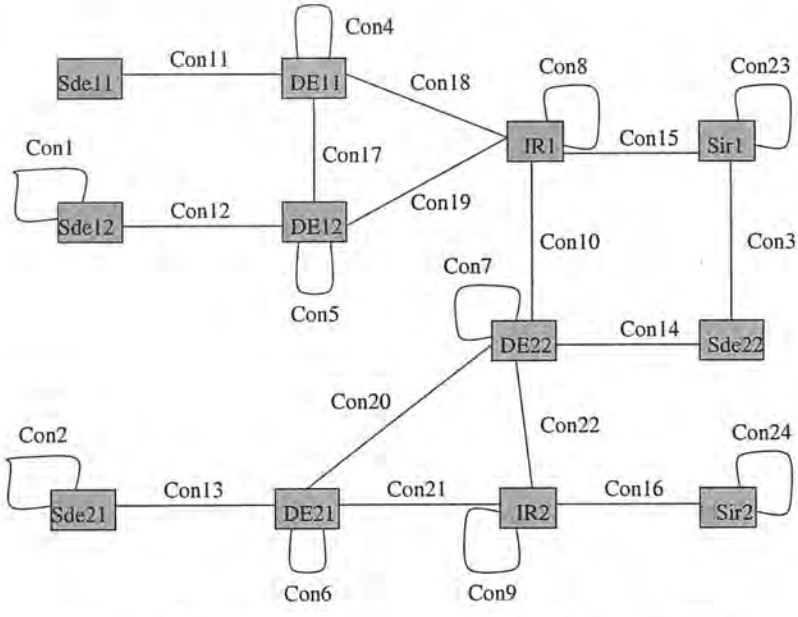


Figure 6.5: The constraint graph of the QLE in (6.25 b)

Table 6.2 shows the results of the variables after resolving source and referential ambiguities. **icon2** and **car2** are a pair of corresponding entities, and **icon2** is the nearest icon to the pointed position, which makes it to be chosen as the pointed entity. The variables generated from the interrogative phrase have value "?", which indicates that their values are not found. As mentioned in Section §6.4.1 and Chapter 5, the reference module does not have to find values for those variables generated from phrases that do not need a referent.

Sde11 = screen	Sde12 = domain	Sde21 = domain
Sir1 = screen	Sir2 = domain	DE11 = icon2
DE12 = car2	DE21 = ?	IR1 = icon2
IR2 = ?	DE22 = ?	

Table 6.2: The results of the variables raised from the QLE in (6.25 b) after the resolution of the CSP

Based on the resolution results, a QLE expression is generated (see (6.26 a)). This expression is sent to the scope binding module, where the corresponding LE expression (see (6.26 b)) is generated. In the execution module, the LE expression is translated into a sequence of PROLOG goals (see (6.26 c)).

(6.26) (a)
```
[list, [pred(is_cheap, domain),
        qterm(interrog, var(X, domain),
              [and, [pred(is_car,domain), var(X, domain)],
                    [pred(is_near, screen),
                          [corresobj(domain),
                                var(X, domain)]),
                          cons(icon2, screen)
                    ]
              ]
              )
        ]
]
```

(b)
```
[list, quant(interrog, var(X, domain),
             [and, [pred(is_car,domain), var(X, domain)],
                   [pred(is_near, screen),
                         [corresobj(domain),
                               var(X, domain)]),
                         cons(icon2, screen)
                   ]
             ],
             [pred(is_cheap, domain), var(X, domain)]
```

```
                    )
                ]
            ]
(c) exec_seq(call_action(is_car, [X], domain),
             call_action(is_near, [corresobj(X, domain),
                                   icon2], screen),
             call_action(is_cheap, [X], domain)
            ).
```

Various processes happen in the execution, which include the computation of the spatial relation `is_near`. As the result of the execution, a domain entity, say `car3`, is found to instantiate the variable `X`. Finally, the referring expression generator produces an appropriate noun phrase denoting the entity `car3`. If the selected attributes are {`Nissan, small, car`}, the referring expression would be `the small Nissan car`. The system's response is displayed on the screen, and this turn of the dialogue between the user and the system is completed (see (6.27)).

(6.27) **User:** Which car near this↖ blue car is cheap?
System: `The small Nissan car.`

6.10 Summary

In this chapter, we presented the prototype IMIG system, which is a multimodal system integrating natural language text input/output, graphic display and pointing gestures. Screen icons and their visual attributes represent domain entities and their domain features.

The system uses an architecture that is similar to a non-multimodal NL system. It has syntactic and semantic analysis components. The parser is based on the ALE environment with an HPSG grammar as the backbone. The main semantic process component, i.e. the reference module, uses the CSP approach explained in Chapter 5.

The execution module has the ability to translate the LE expressions into a list of PROLOG goals and generate appropriate responses in accordance with the results of execution, including referring expressions.

There are several databases in the IMIG system. The world model and the display

model store information about the application domain and the screen respectively. The mapping model stores the mapping relations and knowledge about how to automatically generate the mapping relations. The two remaining knowledge bases are the context model and the general model.

The interface, which is based on Tcl/Tk, facilitates multimodal communication. It displays graphics, mapping relations and dialogues on the screen. It allows the user to type in a natural language sentence with/without pointing actions, or even to use pure screen operations.

Chapter 7

Evaluation

In previous chapters, we discussed the design of the computational model for resolving referring expressions and the implementation of an intelligent multimodal system IMIG with the model as the core. In this chapter, we discuss the evaluation of the model, which examines the usefulness of functions used in the model. Dialogues are selected and their corresponding screen displays are used in the evaluation to provide proper contexts for the multimodal interactions. To avoid implementation limitations, we use the overhearer method in which the IMIG system is indirectly used. The evaluation outcomes are analysed through a serials of statistical tests, whose results tell us that the functions are generally useful in the interaction in both application domains.

7.1 Overview

The essential problems addressed in this thesis are source ambiguities, the ambiguities derived from the situation that *attributes of graphics on the screen could be mentioned in referring expressions, and the graphics could be the referents of the expressions.* Furthermore, we clarify the source ambiguity problems to be the following more specific problems:

P1: *Different parts of the input sentence may have information from different sources. This is a problem for a system assuming that all input information is from the domain.*

P2: *The relation between an anaphoric phrase and its antecedent can be coreferential or quasi-coreferential. The latter has not been discussed in the literature.*

P3: *The identification of source information can be affected by the current context of the interaction.*

P4: *The identification of source information of one part of input can be affected by the source of some other parts.*

To solve these problems, we use a structure explicitly representing the knowledge of graphics and the source information. This makes use of several knowledge bases, (i.e. a display model for the graphics on the screen and their attributes and a mapping model for the relations between domain entities/attributes and screen entities/attributes) and a meaning representation language that marks the source information locally and explicitly. The computational model based on this structure has the following functions targeted at the four problems P1 - P4, respectively.

F1: The model can identify the source of a part of a sentence to be the domain or the screen.

F2: The model can identify the relation between an anaphoric phrase and its antecedent to be coreferential or quasi-coreferential.

F3: The model uses context information to resolve source ambiguities.

F4: The model uses linguistic regularities[1] to resolve source ambiguities.

F5: The model updates the context after some pure screen operations.

Linking to discussion in previous chapters, F1 is related to the thought of treating the ambiguities as source ambiguities. We have identified that the problems are due to lacking information for the source of an entity/attribute, so our resolution model is dedicated to find such missing information.

[1] See Chapter 3.

F2 is related to the discussion in Section §3.8, where quasi-coreference is discussed. A quasi-coreference could be an obstacle if it keeps useful information for source disambiguation away from the resolution process. We have explored a method to handle it in our model (e.g. Rule 3.8).

We believe that there is abundant context information that could be utilised in the source disambiguation process. For example, the information from the antecedent of a quasi-coreference/coreference relation is a piece of context information. Apart from this, RULEs 3.7 to 3.11 are about using inter-sentential context information, and RULEs 3.5 and 3.6 can be viewed as using intra-sentential context information. All of these are used by our resolution model. Therefore, it is reasonable to state that F3 is related to the discussion in several sections in Chapter 3 (Sections §3.6, §3.7, §3.8 and §3.9).

F4 is about using resources from linguistic regularities, and it corresponds to Section §3.5. The heuristic rules that fall into this category are RULEs 3.1 to 3.4, especially RULE 3.4 (the screen head noun rule). Our model also uses such information in the resolution process.

F5 is another function related to the context information, but it aims at handling the context change after pure screen operations.

Comparing to a model without these functions, our resolution model would analyse the input more precisely, and generate more proper responses, so the whole dialogue would be more *natural*. We define that a dialogue is **natural** if it is fluent and free from misunderstanding between its participants. As will be shown, measuring naturalness is the basis of our evaluation.

In general, our experiments aim to evaluate the five functions because they represent the distinctive abilities of our model and the rules we have obtained are specific cases of these functions. If we can prove the usefulness of the five functions, we probably have a clearer idea of what kinds of rule source disambiguation requires. In the experiments, we pick some rules to represent some functions. For example, the screen head noun is tested for F4. In the following, we describe the evaluation in detail.

7.2 Evaluation method

7.2.1 Exploring the usefulness and design limitations of the model

The theme of the thesis is the design of a computational model that is capable of handling source ambiguities. The evaluation is an examination of the design. It tests the functions that are designed to handle source ambiguities, and at the same time, explores limitations of our approach. With the evaluation, we should be able to tell which part of the model actually works, and which parts could be the possible improvement points for our further research.

However, the evaluation has to work on the implementation of the abstract computational model, which means that some considerations from the implementation aspect could affect the evaluation results as well. One important issue in the evaluation is, therefore, how to reduce the effect from the implementation part as much as possible.

7.2.2 Using overhearer method

In chapter 6, we presented the prototype IMIG system. Although it has demonstrated some advance features to handle source ambiguities, it contains various implementation limitations. Its linguistic coverage covers the essential linguistic phenomena for our current exploration of source ambiguities, but this is not sufficient for an interaction to be as natural as that between two persons in all aspects. The graphic display of the IMIG system is adequate for the demonstration purpose, but is still too primitive to be an elegant visualisation tool. The robustness of a demonstration system in the hands of a real user is always a challenge, and this is true in the IMIG system. In addition, the subject of our evaluation is not the IMIG system, but rather the computation model behind the system. Therefore, although people have developed various methods to evaluate human computer interaction systems [Dix, 1998, chapter11], we need a special method to examine the model without being affected by the limitations of the IMIG system.

We use the method we call the *overhearer method*, whose basic idea is that the subjects are not the direct participants of dialogues, but the third party who observe the dia-

logues between a user and the system. Depending on the purpose of experiments, their tasks vary. In our evaluation, the subjects are required to make judgements about the naturalness of the dialogues. The reasons for using this method are:

- The overhearer method makes it possible to involve the IMIG system indirectly so that we can show relevant aspects of the IMIG system to the subjects, but conceal the irrelevant parts. This helps to avoid the effect of implementation limitations on their judgements and to keep subjects concentrating on the main issues.
- The overhearer method is not entirely odd to the science paradigm. Similar methods have been used in other research. For example, [Cox et al., 1999] describes studies about vicarious learning . They aim to demonstrate that people could learn about the topics in dialogues even in vicarious situations. Although we are aiming at different tasks (evaluation vs learning), we do share the same insight, that is, dialogues are not merely the property of their participants, and other people can grasp relevant information by observing dialogues.

We acknowledge that the overhearer method has limitations. The major disadvantage is that subjects might not receive adequate information about the dialogue to make appropriate judgements. However, based on the following reasons, we believe that this disadvantage can be overcome.

- The evaluation environment is multimodal, where both linguistic dialogues and the corresponding graphic displays provide rich context information to the overhearer.
- The evaluation environment is a controlled situation. We can select dialogues that are easy to understand.
- We interviewed the subjects after they finished the naturalness judgement so that we could identify any misunderstanding from the subjects (see Section §7.2.4).

7.2.3 Comparing two groups

To evaluate the effectiveness of the functions, we compare two groups of dialogue, where one is assumed to be generated from a resolution model that always has the functions, whereas the other is assumed to be generated from a comparable model that always does not have the functions. The comparable model obviously does not exist, so all the dialogues generated from it are just fictitious and imaginary for the purpose of comparison. The evaluation is supposed to show whether or not the naturalness of the dialogues in the first group is significantly higher than that of the dialogues in the second group. To build the comparison between the dialogues in the two groups, whenever there is a guess in processing the dialogues in the second group, the comparable model always picks up the different result to the one generated from the resolution model. Further detail about the comparison can be seen in Sections §7.3.3 to §7.3.7.

Each dialogue has a sentence or part of a sentence used to test the function that the dialogue aims to test. That sentence or the part of the sentence is called the **test point** of the dialogue. The comparison of two groups of dialogues is on their test points.

7.2.4 Evaluation procedure

The evaluation is organised in the following stages:

1. *Preparation:* to test the usefulness of F1 to F5, a number of dialogues between a user and the IMIG system were selected and polished to avoid unwanted side-effects from implementation. They were presented in a way that the subjects could perceive all the relevant information easily.

2. *Examination:* the subjects were asked to read the dialogues, watch the screen displays, and then write down their judgements of the naturalness of the dialogues.

3. *Interview:* after the examination, the subjects were asked to review their decisions. The commonly asked question during the interview was "*why do you think this part of text is unnatural?*" The main aim of the interview was to record

the reasons behind their judgements, and identify whether or not the obtained results were relevant to our concern and free from misunderstanding.

4. *Analysis:* when enough data are collected, we used statistical methods to analyse the data.

Three interesting issues arise from the above stages. They are *test dialogue selection and polishing* (Section §7.3), *presentation* (Section §7.4) and *result analysis* (Section §7.5).

7.3 Test dialogue selection and polishing

By using the overhearer method, the subjects do not directly interact with the IMIG system during the evaluation. Instead, they read selected dialogues, watch the corresponding screen displays and make judgements. In this situation, the dialogues used in the evaluation are crucial, so selecting test dialogues deserves special attentions.

7.3.1 General guidelines for selection

There is no existing corpus to select dialogues directly, so the test dialogues have to come from the imaginary ones based on the scenarios we used during the development of the resolution model.

Some general guidelines are set up to make sure that the selected dialogues have the necessary information, easily understood and can help us to achieve the evaluation goal.

The first one is that **the dialogues should include both linguistic forms and graphic displays.**

The dialogues used in the evaluation are from a multimodal environment involving not only conversations in linguistic forms, but also the accompanying screen displays. Without the screen displays, the visual part of the dialogue context would be missing. Consequently, a selected dialogue should have both the linguistic part and the corresponding screen display of the conversation.

However, graphic displays are only used to provide the context of the dialogues. They do not contain the test points. All the test points are in the linguistic part of the conversations. This is because the resolution model we build is still a language processing module.

The second guideline is that **the dialogues should come from two different domains.**

One of the ultimate goals of the resolution model is to achieve domain independence. Two application domains are considered in developing the model. One domain is the *Car Selection domain*, and the other is the *Room Arrangement domain* (see Chapter 1).

The most significant difference between these two domains is that the domain entities in the car selection domain have no domain spatial relation among them, whereas the domain entities in the room arrangement domain do. Therefore, if a spatial relation is involved in an interaction about the car selection domain, the relation must have the screen source. No such certainty exists in an interaction about the room arrangement domain. This is why we call the car selection domain a *non-spatial domain*, whereas the room arrangement domain a *spatial domain*. Detailed descriptions about these two domains were given in Chapter 1.

To test the influence of the domain difference on the model, test dialogues should come from both domains for each function, and their numbers should be balanced.

The third guideline is that **the dialogues should be selected in pairs and in similar context.**

A comparison based evaluation, intuitively, requires a pair based selection. For each function, we chose one dialogue from the model with that function and one from the model without it. In the case of the latter, the responses are based on our manual interpretation of the user's input.

To test the five functions in comparison, we need 5 pairs (i.e. 10 dialogues). If considering domain independence (i.e. functions are tested in both two domains), the smallest number of the selected dialogues is 10 pairs (i.e. 20 dialogues).

A dialogue contains a lot of information. Many factors could affect its naturalness.

To make sure that the obtained results are really about the test point, the differences between the pair of dialogues have to be reduced to a minimum. Consequently, the two dialogues in a pair are the same except the exact test point.

Apart from the selection, the dialogues also need to be polished to get rid of the effects on their naturalness of the limited linguistic coverage of the parser and rigid functionality of the interface.

7.3.2 General guidelines for polishing

The implemented system, IMIG, has a reasonable linguistic coverage, but, as mentioned in Chapters 1 and 6, it cannot handle some phenomena. For example, it cannot deal with ellipsis and one anaphora; its responses sometimes sound rigid because it only has a very basic referring expression generation module and virtually no natural language generation module at all; and it assumes that the user would never input any assertion (declarative sentence) during the interaction.

These limitations are irrelevant to the design of the computational model. If controlled carefully, they should not severely damage the interaction. However, they do pose problems to our evaluation, since they could affect human decisions on the naturalness of dialogues, which is a core factor of our evaluation.

Aiming at avoiding the effect of implementation limitations and enabling the subjects to concentrate on the design aspects, we polished selected dialogues before using them in the evaluation. Of course, polishing is done carefully not to conceal any design fault. In the rest of this section, we are going to present two general guidelines and some types of polishing with fragments of dialogues as examples. The full range of polishing can be found in Appendex B

Each selected dialogue serves a test purpose, which is achieved by using some particular aspects or components of the dialogue as a probe to form a test point. The first guideline is that *polishing should never be used on test points.*

The implementation limitations mainly concern the syntactic aspects of the dialogues. Therefore, the second guideline is that *polishing should never change the meaning, or at least the core meaning, of the sentences.*

In the rest of this section, some dialogue fragments are presented to illustrate the details of polishing. The selected dialogues in full length can be found in Section §7.3.3 to Section §7.3.7. The full range of polishing can be found in Appendix B.

(7.1) **User:** Is this $\nwarrow_{(c)}$ car a five-door hatchback? (c)
System: `Yes, it is.` (d)
User: What about the green car near it then? (e)
System: `No, that one is a three-door hatchback.` (f)

(7.1) is an example of polishing on ellipsis. Before polishing, (e) reads "`Is the green car near it a five-door hatchback?`". The meaning of the polished sentence is the same as the original one. The phrase "`the green car near it`" in (e) is the test point, which is kept untouched. A comparison between (7.4) and (7.5) in Section §7.3.3 gives a clearer example.

(7.1) is also an example of polishing on one anaphora. In (f), the phrase "`that one`" is used in the system's reply, whereas the IMIG system should generate a referring expression like "`the green car`" according to its rules in the referring expression generation module. Again using either "`that one`" or "`the green car`" does not make any difference to the sentence meaning, but the former is more likely to be used by human in this situation.

(7.2) **User:** How much is the green car? (a)
System: `14,200 pounds.` (b)
User: It is too expensive. Remove it from the screen. (c)
System: `The green car has been removed from the screen.` (d)

The test point in (7.2) is the pronoun "`it`" in the second sentence of (c). This dialogue is an example of assertion sentence polishing. The IMIG system assumes no assertion in the interaction, so the first sentence in (c) would not appear in the interaction. However, that sentence gives the reason why the user wants to remove the green car, which makes the dialogue smoother, and it does not affect any of the system's decision on the referents of phrases (e.g. "`it`") or the sources of sentence components.

(7.3) **User:** Give me more information about it. (h)
System: `You already know that the price of this car is 14,200 pounds and it is a green car. The other information is ...` (i)

In (7.3), polishing happens in (i), whereas the test point is the pronoun "`it`" in (h). In reality, the system's reply looks like "`It is 4,200 pounds. It is a green car. It is ...`" Although the pre-polished reply serves the purpose of giving the user more information about the car, its style is so bad that it might cause the subjects to mark the whole dialogue as unnatural. Therefore, polishing is needed.

In summary, polishing is used to overcome implementation limitations so that they will not affect the naturalness of the selected dialogues. Polishing has been carefully performed to preserve the sentence meanings and not to conceal design faults. From Section §7.3.3 to Section §7.3.7, we will talk about how to select the dialogues to test the validity of some design decisions of the computational model. All the dialogues shown there have been polished. The corresponding unpolished dialogues are given in Appendix B.

7.3.3 Selecting dialogues for testing F1

The F1 function says that "*the model can identify the source of a part of a sentence to be the domain or the screen*". To test the function, one model is assumed to have the function, whereas the other model does not. This means that the latter would always assume that the information is from one database. To achieve the contrast between the two following dialogues, we assume that the model without F1 always resolves the information to come from the domain.

(7.4) and (7.5) are a pair of the selected dialogues in the car selection domain for testing F1. The test points are the phrase "`the green car near it`" in (e). In (7.4), the source of "`green`" in (e) is recognised to be the screen, so the intended referent of the phrase "`the green car`" is found and the relevant information is presented in (f). In (7.5) the source of the word "`green`" in (e) is assigned to be the domain, and no domain entity is found to be the intended referent of the phrase "`the green car`".

As a result, an error message in (f) is generated.

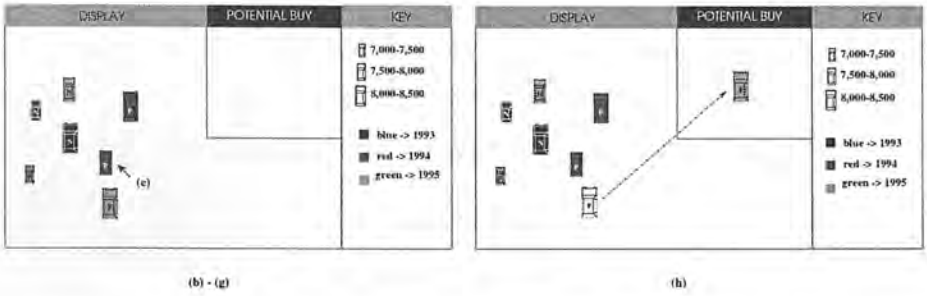


Figure 7.1: The screen display for (7.4) of the Car Selection domain

(7.4)
User: Is there a silver car in your stock? (a)
System: `Yes, we have quite a few. Here all the silver cars are displayed on the screen.` (b)
User: Is this $\nwarrow_{(c)}$ car a five-door hatchback? (c)
System: `Yes, it is.` (d)
User: What about the green car near it then? (e)
System: `No, it isn't. It is a three-door hatchback.` (f)
User: Move the green car to the potential buy area. (g)
System: `The green car has been moved.` (h)

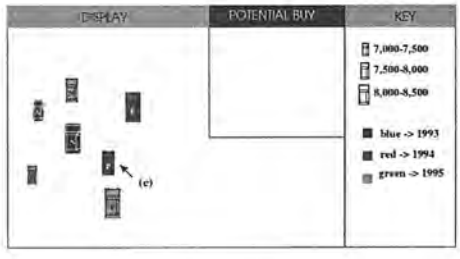


Figure 7.2: The screen display for (7.5) of the Car Selection domain

(7.5)
User: Is there a silver car in your stock? (a)
System: `Yes, we have quite a few. Here, all the silver cars are displayed on the screen for you.` (b)
User: Is this $\nwarrow_{(c)}$ car a five-door hatchback? (c)
System: `Yes, it is.` (d)
User: What about the green car near it then? (e)
System: `In our stock, there is no green car near that five-door hatchback.` (f)

7.3.4 Selecting dialogues for testing F2

The F2 function says that "*the model can recognise coreferential and quasi-coreferential relations between anaphoric phrases and their antecedents*". If a model does not have this function, it would take a quasi-coreferential relation as a coreferential relation.

(7.6) and (7.7) are a pair of dialogues from the car selection domain for testing F2. The test point is the relation between the second pronoun "`it`" in (c) and its antecedent, "`the green car`" in (a). The model behind (7.6) recognises the quasi-coreferential relation and assigns the icon of the green car to be the intended referent of the second "`it`" in (c). As a result, the response in (d) is generated. (7.7) shows the situation without F2, where the intended referent of the second "`it`" in (c) is resolved to be a green car rather than the icon of a green car. Since a domain car cannot be used in the `remove` operation in the car selection domain, the error message given in (d) is generated.

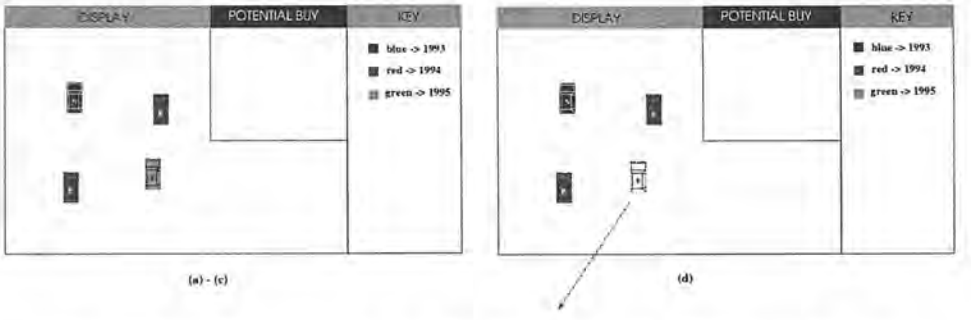


Figure 7.3: The screen display for (7.6) of the Car Selection domain

(7.6) **User:** How much is the green car? (a)
System: `14,200 pounds.` (b)
User: It is too expensive. Remove it from the screen. (c)
System: `The green car has been removed from the screen.` (d)

(7.7) **User:** How much is the green car? (a)
System: `14,200 pounds.` (b)
User: It is too expensive. Remove it from the screen. (c)
System: `Remove can only be used on icons, but your phrase ''it'' refers to a car.` (d)

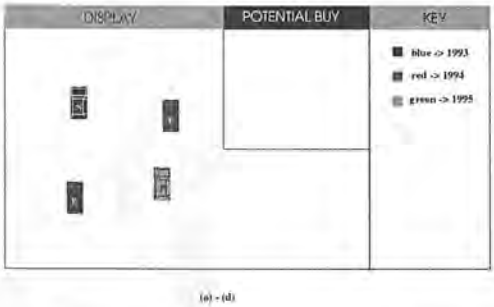


Figure 7.4: The screen display for (7.7) of the Car Selection domain

7.3.5 Selecting dialogues for testing F3

The F3 function says that "*the model uses the context information to resolve the source ambiguities*". The test point is that previous resolutions would not affect the current resolution in a model without the function, whereas they would in a model with the function.

(7.8) and (7.9) are a pair of selected dialogues from the Room Selection domain. They are identical up to (f). In (c), they both interpret the source of the word "`above`" to be the screen because only the screen source can fit into the user's input. However, difference appears in the resolution of the source of the word "`above`" in (e). (7.8) is from the model with F3, which knows from the context that the word "`above`" was used in (c) with the screen source. Using the preference that the source of the word "`above`" in (e) is also the screen, the system moves the table to the place shown in Figure 7.5. However, the model behind (7.9) could not use context information, so for the purpose of comparison, we assume that it makes a decision that the source of the word "`above`" is the domain. As a result, the system moves the table to the place shown in Figure 7.6.

(7.8)
User: What is the size of this $\nwarrow_{(a)}$ table? (a)
System: `53cm x 79cm.` (b)
User: What is the distance between this $\nwarrow_{(c)}$ cabinet and the wall above it? (c)
System: `You mean this` $\nwarrow_{(d)}$ `wall? The distance is 60cm.` (d)
User: Move the table to the place above the cabinet. (e)
System: `The table has been moved.` (f)

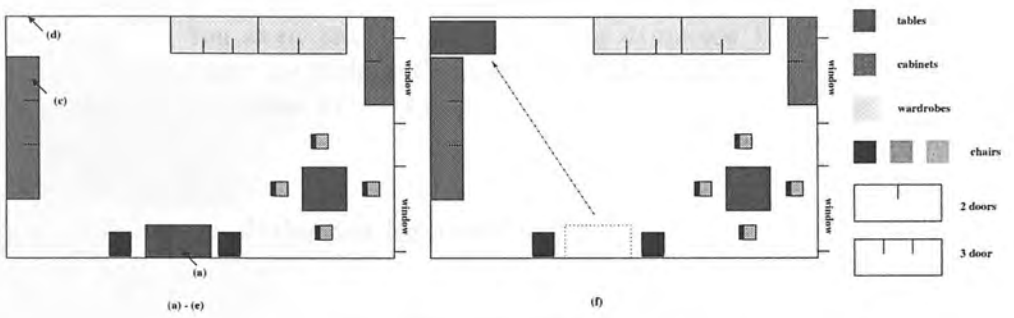


Figure 7.5: The screen display for (7.8) of the Room Arrangement domain

Figure 7.6: The screen display for (7.9) of the Room Arrangement domain

(7.9) **User:** What is the size of this $\nwarrow_{(a)}$ table? (a)
System: `53cm x 79cm.` (b)
User: What is the distance between this $\nwarrow_{(c)}$ cabinet and the wall above it? (c)
System: `You mean this` $\nwarrow_{(d)}$ `wall? The distance is 60cm.` (d)
User: Move the table to the place above the cabinet. (e)
System: `You mean like this` $\nwarrow_{(f)}$`?` (f)

7.3.6 Selecting dialogues for testing F4

The F4 function says that "*the model uses linguistic regularities to resolve the source ambiguities*". This time we test the most frequently used rule, the screen head noun rule (see Chapter 3). This rule says that the modifiers of a phrase and the intended referent of the phrase would come from the screen if the head noun of the phrase comes from the screen. A model with F4 would assign the sources of the modifiers and the intended referent of a phrase to be the screen if the screen head noun rule is satisfied, whereas a model without F4 would have to guess. Here we assume that the second model makes the decision that the source is the domain.

(7.10) and (7.11) are a pair of dialogues from the car selection domain. The test point in both dialogues are the phrase "`the green icon`" in (g). In (7.10), because the model uses the screen head noun rule, the source of the modifier "`green`" is resolved to be the screen, so the icon with green colour is moved. However, (7.11) is from a model without F4, so for the purpose of contrast, the source of the word "`green`" is assigned to be the domain, so the icon representing a green car is moved.

(7.10) **User:** What is the colour of this $\nwarrow_{(a)1}$ car and this $\nwarrow_{(a)2}$ car? (a)
System: `The first car is green, and the second one is white.` (b)
User: Is the green car a five-door hatchback? (c)
System: `Yes, it is.` (d)
User: How about the white car? (e)
System: `No, it is not. It is a saloon.` (f)
User: Move the green icon to the potential buy area. (g)
System: `You mean this` $\nwarrow_{(h)}$ `icon?` *[Action: The icon is moved.]* `It has been moved.` (h)

(7.11) **User:** What is the colour of this $\nwarrow_{(a)1}$ car and this $\nwarrow_{(a)2}$ car? (a)
System: `The first car is green, and the second one is white.` (b)

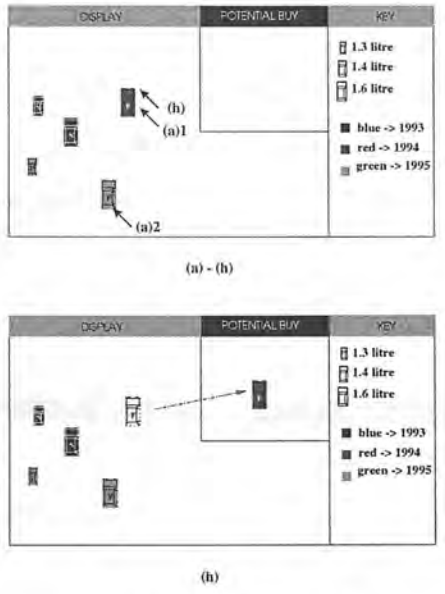


Figure 7.7: The screen display for (7.10) of the Car Selection System

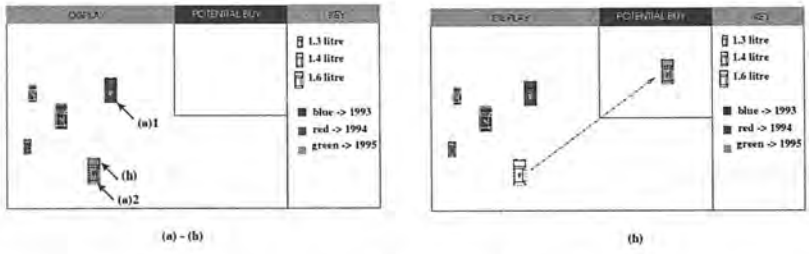


Figure 7.8: The screen display for (7.11) of the Car Selection System

User: Is the green car a five-door hatchback? (c)
System: `Yes, it is.` (d)
User: How about the white car? (e)
System: `No, it is not. It is a saloon.` (f)
User: Move the green icon to the potential buy area (g)
System: `You mean this` $\nwarrow_{(h)}$ `icon?` *[Action: The icon is moved.]* `It has been moved.` (h)

7.3.7 Selecting dialogues for testing F5

The F5 function says that "*the model updates context after some pure screen operations*". The test point here is that if the context is changed after some pure screen operation, say moving an entity, the subsequent referent resolution should reflect this

change.

(7.12) is a dialogue from a model with F5. The model recognises the context change from the `move` operation in (g), and resolves the intended referent of the pronoun "`it`" in (h) to be the car whose icon was just moved. In contrast, (7.13) shows a dialogue from a model without F5. The context is not affected by the "`move`" operation, so the model still resolves the intended referent of the pronoun "`it`" in (h) to be the white car mentioned in (e).

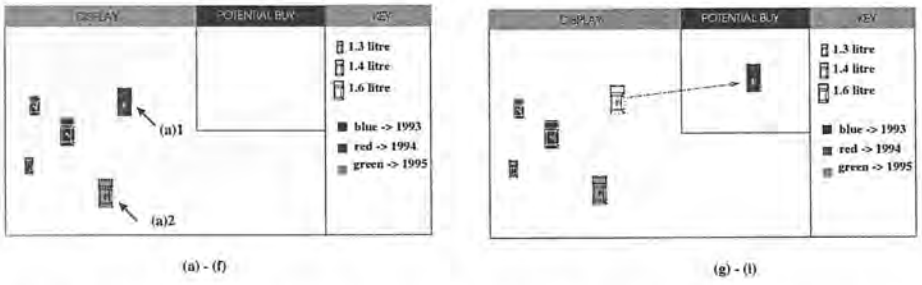


Figure 7.9: The screen display for both (7.12) and (7.13) of the Car Selection domain

(7.12) **User:** What is the colour of this $\nwarrow_{(a)1}$ car and this $\nwarrow_{(a)2}$ car? (a)
System: `The first car is green, and the second one is white.` (b)
User: How much is the green car? (c)
System: `14,200 pounds.` (d)
User: How about the white car? (e)
System: `13,000 pounds.` (f)
Action: *After reading the answer of the system, the user moves the icon of the green car to the potential buy area (see Figure 7.9 (g) - (i)).* (g)
User: Give me more information about it. (h)
System: `You already know that the price of this car is 14,200 pounds and it is a green car. The other information is ...` (i)

(7.13) **User:** What is the colour of this $\nwarrow_{(a)1}$ car and this $\nwarrow_{(a)2}$ car? (a)
System: `The first car is green, and the second one is white.` (b)
User: How much is the green car? (c)
System: `14,200 pounds.` (d)
User: How about the white car? (e)
System: `13,000 pounds.` (f)
Action: *After reading the answer of the system, the user moves the icon of the green car to the potential buy area (see Figure 7.9 (g) - (i)).* (g)
User: Give me more information about it. (h)

System: `You already know that the price of this car is 13,000 pounds and it is a white car. The other information is ...` (i)

7.4 Presentation

By presentation, we mean the method used to show the selected dialogues for subjects' examination. It is important because the IMIG system is not used directly in the evaluation.

There is no important theoretical principle as to how to present the selected dialogues and their visual displays. What we have are a few practical issues.

The presentation has to be able to deliver colour information because each dialogue is accompanied by one or more screen displays. Colour and other visual display information are critical for both interaction and evaluation. For this reason an on-line questionnaire or a colour printed paper-based questionnaire is a good choice.

However, both of them have limitations in our evaluation. To test all five functions in two different application domains, 20 dialogues and their visual displays are selected. To put them in a readable paper-based questionnaire is at least 22 pages long, which could drive potential subjects away.

An on-line questionnaire also has its problem. Our evaluation asks subjects to mark the unnatural part of the dialogues, and it is very difficult to build a robust display system which allows the subjects to circle or mark any part of the presentation on the screen.

Therefore, we decided to combine the two methods. The presentation consists of a web page showing the selected dialogues and their visual displays, and a paper based answer sheet for the subjects to mark on.

7.4.1 Presenting the dialogues and visual displays in web pages

Figure 7.10 shows a snapshot of the web page read by subjects during the evaluation. We use frame technique to divide the browser's window into two parts. The part at the left side is the content window listing the URL links to the dialogues and their accompanied screen displays. Subjects can click the appropriate links to load a dialogue and its screen displays into the other part of the browser's window. A warning message about selecting the right dialogue with the right displays is constantly shown above the dialogues to avoid any mismatch between dialogues and screen displays.

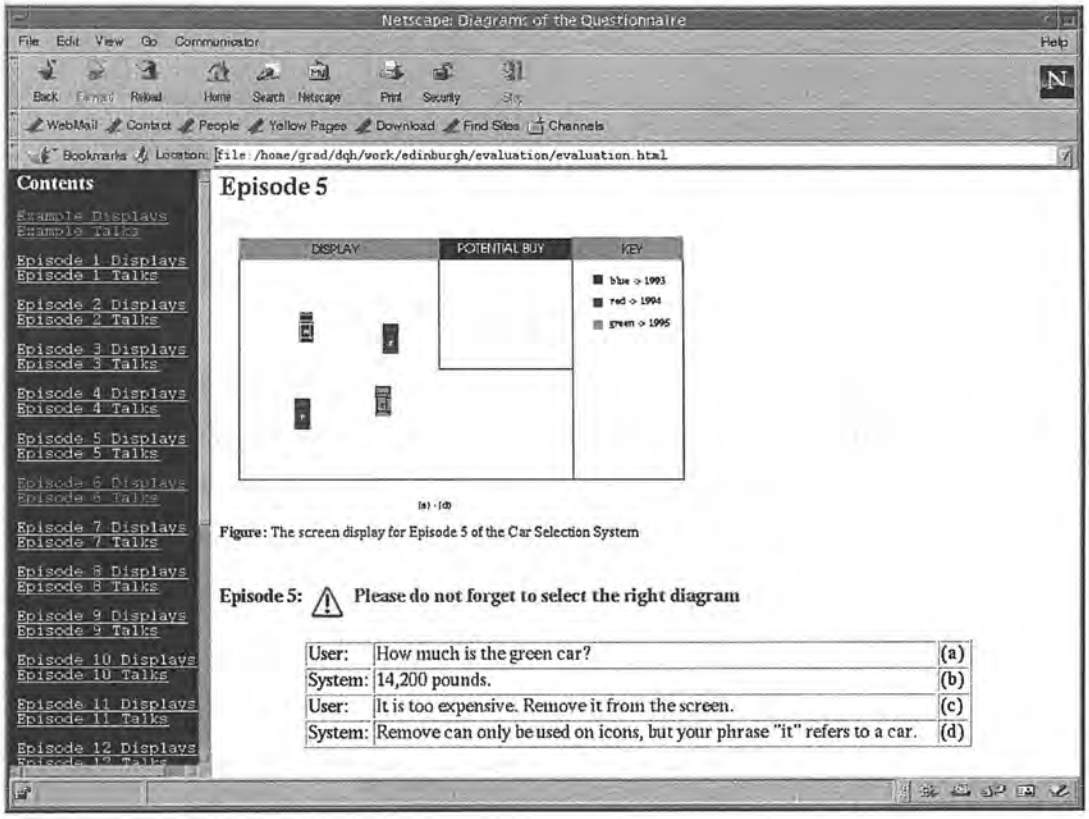


Figure 7.10: A snapshot of the on-screen presentation web page

7.4.2 Marking the verdicts on the answer sheet

The answer sheet gives the remaining information about the experiment. At the beginning of the sheet, it has instructions about the aim of the experiment, the role of subjects and their tasks. The reason for us to put the instructions in the answer sheet

rather than on the screen is that we believe people generally are more comfortable with reading from a paper than from a screen. The detailed instructions can be found in Appendix C.

After the instructions, the text parts of the selected dialogues are presented for subjects to mark on. The subjects are expected to make a judgement when their memories about the dialogues are still fresh. Therefore, immediately after the text part of each dialogue, two questions are asked. The first question accesses the subjects' impression on the naturalness of the dialogue, which has a scale with five levels:

1 - unnatural

2 - fairly unnatural

3 - so-so

4 - fairly natural

5 - natural

The subjects are asked to read the dialogues and the screen displays carefully. If they think the whole dialogue is free from misunderstanding, they should circle 3 to 5 according to how natural they think the dialogue is. If any part of the dialogue has some misunderstanding, they should circle 1 or 2 according to how bad they think the misunderstanding is.

The second question should be answered if the subjects think the naturalness of a dialogue is either 1 (i.e. *unnatural*) or 2 (i.e. *fairly unnatural*). They are asked to circle any part in the dialogue that seems unnatural. For example, (7.14) is an example of a marked dialogue on the answer sheet, where the dialogue is thought to be unnatural and the unnatural part is the phrase "`the green icon`" in (f), which has been circled.

(7.14)
User: What is the colour of this $\nwarrow_{(a)}$ car? (a)
System: `It is green.` (b)
User: Is its insurance group lower than that of this $\nwarrow_{(c)}$ red car? (c)
System: `Yes, its insurance group is lower.` (d)
User: Move the green car to the potential buy area. (e)
System: (`The green icon`) `has been moved.` (f)

Naturalness: (1) 2 3 4 5

If you choose 1 or 2, please circle on the dialogue the part(s) that seem unnatural:

7.4.3 Other issues

To avoid sequence bias on the results, we arbitrarily selected a time and changed the order of the dialogues once during the experiment. Some subjects (4 people) use one sequence of dialogues, and the rest (10 people) use the other sequence of dialogues.

Table 7.1 is a summary of the roles of the 20 dialogues in the evaluation. The details of the 20 dialogues are shown in Appendix B. In the table, each dialogue is marked by its sequence number in the appendix. For example, B.2 and B.8 are two dialogues shown in Section §B.2 and §B.8. They are also the pair of dialogues from the Room Arrangement domain for testing F1. B.2 is from a model with F1, whereas B.8 is from the one without.

	Room with	**Room without**	**Car with**	**Car without**
F1: annotation	B.2	B.8	B.1	B.14
F2: coref & quasi-coref	B.18	B.11	B.10	B.5
F3: context	B.6	B.17	B.7	B.16
F4: regularities	B.4	B.20	B.15	B.9
F5: pure screen	B.3	B.13	B.12	B.19

Table 7.1: The summary of the tests in the dialogues

7.5 Results analysis and discussion

7.5.1 Results descriptions

The subjects

We recruited 14 subjects, among whom 13 were native English speakers (one from USA, and the others from Britain) and the remaining one was from Germany but spoke nearly native English. All of them had used a computer before, but none of them had any experience with the IMIG system. Eight of them were postgraduate

students, and the rest were undergraduates.

Data

We used all the 14 answer sheets, so in total there are 280 naturalness scores. Table 7.2 and Figure 7.11 are a summary of the distribution of the naturalness scores.

Value Label	Value	Total Frequency (No/Percent)	Using an F Frequency (No/Percent)	Not Using F Frequency (No/Percent)
unnatural	1	57/20.4%	7/5.0%	50/35.7%
fairly unnatural	2	62/22.1%	8/5.7%	54/38.6%
so so	3	35/12.5%	20/14.3%	15/10.7%
fairly natural	4	70/25.0%	57/40.7%	13/9.3%
natural	5	56/20.0%	48/34.3%	8/5.7%
Total		280/100.0%	140/100.0%	140/100.0%

Table 7.2: The summary of the naturalness scores

Table 7.2 shows the distribution of the five naturalness values among 280 scores using both absolute quantities and percentages. The last two columns show the data for the model WITH a function (Using an F) and WITHOUT any function (Not Using F) respectively. Figure 7.11 shows the same information in graphics. The figure clearly demonstrates that the majority of the naturalness scores of the model WITHOUT a function (Not Use F) are *unnatural* and *fairly unnatural*, whereas the majority of that of the model WITH a function (Use F) are at *natural* and *fairly natural* levels. This has probably demonstrated that dialogues from the model with a function are more natural than those from the model without any function. Another diagram, Figure 7.12, provides the reader a detailed comparison of the naturalness distribution under each evaluation condition. The data are plotted according to the pairs illustrated in Table 7.1.

Normality of the Data

Before an appropriate statistical method can be selected for analysing the data, we need to examine the normality of the obtained data to see if the individual responses under a given experimental condition are of normal distribution. Given there are three

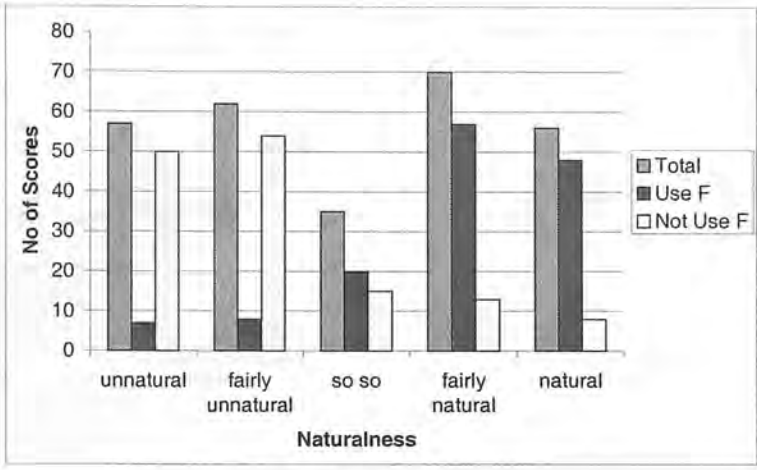


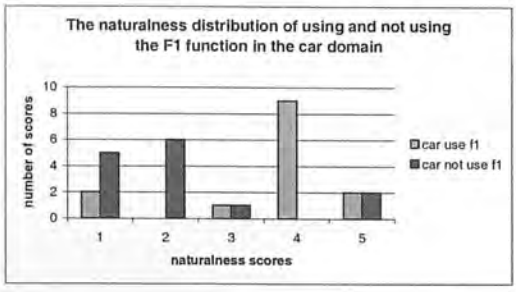
Figure 7.11: The distribution of the naturalness scores

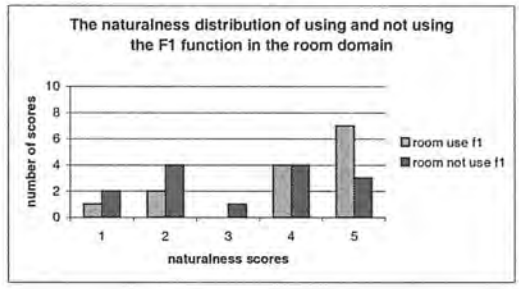
independent variables that contribute the naturalness of the dialogues (i.e. one represents the two domains, another represents the five functions and the third represents having or not having a function), there are totally 20 different conditions (2 domains $\times$ 5 functions $\times$ 2). We find that none of the distributions of the 20 conditions about naturalness vs variables is normal, which motivates the use of non-parametric analysing methods (the detailed distribution diagrams are given in Figure 7.12).

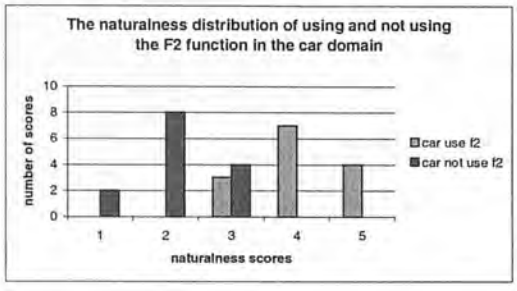
Statistical tests

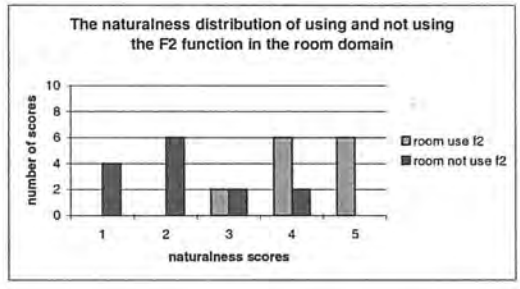
Our evaluation compares two groups of dialogues, one from a model with the five functions and the other from a model without. All the subjects read both groups of dialogues, watch all the screen displays and answer the same questionnaire. The non-parametric statistical method that suits this situation is Wilcoxon Matched-Pairs Signed-Ranks Test [Hatch and Lazaration, 1991]. The test compares the two groups by weighing the differences between pairs of scores. The *null hypothesis* assumes that *the two groups are from the same distribution, therefore the signed rank difference should be statistically insignificant.* Based on the obtained significance level, we can accept or reject the null hypothesis. The significance is calculated by subtracting from the total probability (i.e. 1) the probability (P) of the obtained Z (see Table 7.3).

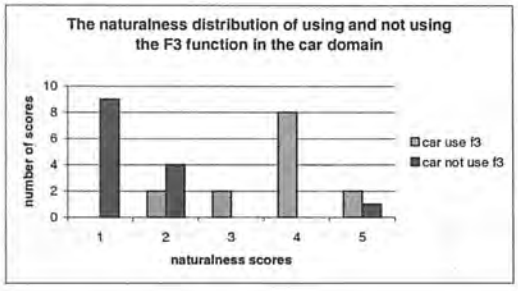
We performed a series of the Wilcoxon Matched-Pairs Signed-Ranks Tests concerning

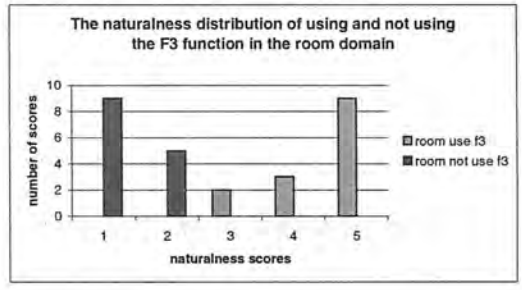


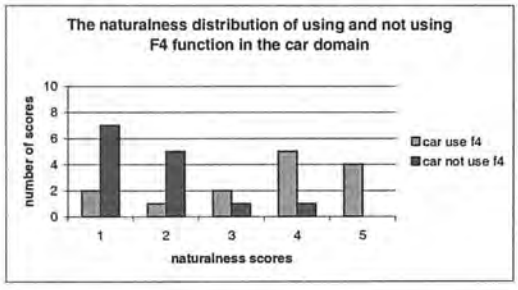


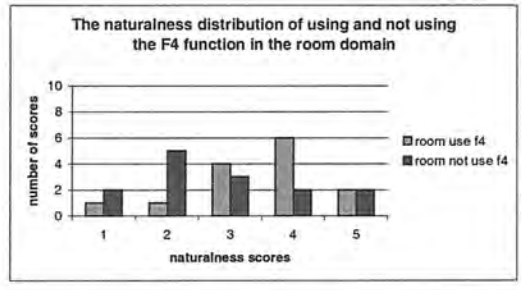


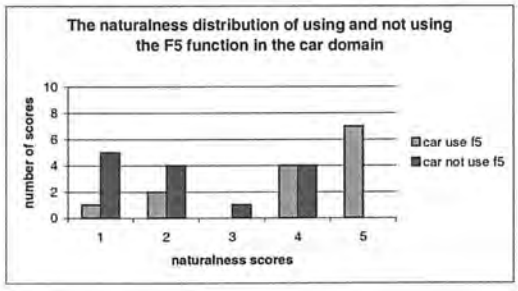












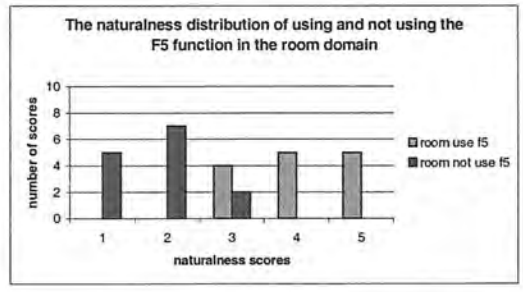


Figure 7.12: The distribution comparison of the naturalness scores under each evaluation condition

the usefulness of F1, F2, ..., and F5. The experiments are carried out under three categories: analysing dialogues from the car selection domain, the room arrangement domain and both. The results for dialogues from both domains are shown in Table 7.3. "*GT*" and "*LT*" stand for "*greater than*" and "*lower than*" respectively. The first column indicates which function(s) (i.e. F1 to F5 for row 1 to 5, and all the functions for row 6) is(are) presented in each row. The last column shows the confidence level in rejecting the null hypotheses. For example, the 2-Tailed P in the first row (0.010) indicates that we have about 99% confidence in rejecting the null hypothesis concerning F1. Using conventional threshold 0.05, we can quite confidently say that the naturalness of the dialogues from models with and without F1 are significantly different. If considering the fact that most dialogues from the model with F1 have high naturalness scores, then it is very likely that the naturalness of the dialogues from a model with F1 is significantly higher than that from a model without. This indicates that F1 is useful.

For the same reason, all other null hypotheses in the table are rejected with high confidence, which indicate that the functions are useful.

The data in other columns are used for calculating the 2-Tailed P. They are useful for checking the correctness of the results in the last column, but can be safely ignored if the reader is just interested in the evaluation results.

Naturalness by	**Mean Rank/Pairs of using GT not using**	**Mean Rank/Pairs of using LT not using**	**Ties**	**Z**	**2-Tailed P**
F1	12.39/18	10.06/5	5	-2.585	.010
F2	13.00/25	0.00/0	3	-4.373	.000
F3	13.96/25	2.00/1	2	-4.407	.000
F4	12.00/20	12.00/3	5	-3.102	.002
F5	12.23/20	4.25/2	6	-3.831	.000
All F	62.03/108	40.09/11	21	-8.298	.000

Table 7.3: The Sign-Ranks Test results based on the judgements in both domains

Table 7.4 and Table 7.5 are the test results when only the dialogues from the car selection domain or those from the room arrangement domain are considered. They use the same notation as that in Table 7.3.

Naturalness in car by	Mean Rank/Pairs of using GT not using	Mean Rank/Pairs of using LT not using	Ties	Z	2-Tailed P
F1	7.40/10	2.00/2	2	-2.746	.006
F2	6.50/12	0.00/0	2	-3.059	.002
F3	7.00/11	1.00/1	2	-2.981	.003
F4	7.08/12	6.00/1	1	-2.761	.006
F5	6.44/8	1.75/2	4	-2.446	.014
All F	32.15/53	11.00/6	11	-6.182	.000

Table 7.4: The Sign-Ranks Test results based on the judgements in the car selection domain

Naturalness in room by	Mean Rank/Pairs of using GT not using	Mean Rank/Pairs of using LT not using	Ties	Z	2-Tailed P
F1	5.69/8	6.83/3	3	-1.111	.266
F2	7.00/13	0.00/0	1	-3.180	.002
F3	7.50/14	0.00/0	0	-3.296	.001
F4	5.25/8	6.50/2	4	-1.478	.139
F5	6.50/12	0.00/0	2	-3.059	.002
All F	30.46/55	30.90/5	10	-5.599	.000

Table 7.5: The Sign-Ranks Test results based on the judgements in the room arrangement domain

The results obtained from the car selection domain (see Table 7.4) is almost the same as those in Table 7.3. But among the results from the room arrangement domain (see Table 7.5), only $F2$, $F3$, $F5$ and $allF$ rows achieve statistically significant difference, whereas $F1$ and $F4$ rows do not have confidence high enough (i.e. the confidence for $F1$ is 73.4% and that for $F4$ is 86.1%) to reject the corresponding null hypotheses.

In summary, our evaluation tells us that, in most evaluation conditions, the overall naturalness of the dialogues using the five functions is significantly higher than that of the dialogues not using the functions (see Table 7.6). Therefore, our evaluation has demonstrated that those functions are useful in constructing more coherent dialogues.

Test according to	**The Car Selection Domain**	**The Room Arrangement Domain**	**Both Domains**
F1	significant	**insignificant**	significant
F2	significant	significant	significant
F3	significant	significant	significant
F4	significant	**insignificant**	significant
F5	significant	significant	significant
All F	significant	significant	significant

Table 7.6: The degree of difference in the naturalness values of dialogues using a function and not using a function in the car selection domain, the room arrangement domain and both

7.5.2 Discussion

General remarks

Generally, the evaluation results enable us to believe with high confidence that the functions are useful, at least for the test dialogues. The high confidence is based on the significant P value

- when considering the naturalness judgements on applying each and all the functions in both domains;
- when considering the naturalness judgements on applying F1 to F5 individually in the car selection domain;

- when considering the naturalness judgements on applying F2, F3 and F5 individually in the room arrangement domain.

However, we found no significant difference in the naturalness judgements on applying F1 or F4 in the room arrangement domain. In the following, we try to explain what causes the poor results.

Each selected dialogue has a test point, which could affect the value of the naturalness of the dialogue. When the dialogue is selected and polished, its naturalness score has an expected range. For example, (7.12) and (7.13) are a pair of dialogues testing the usefulness of F5 in the car selection domain. Because we think that (7.12), which is generated by the resolution model, has higher naturalness score, its expected naturalness score is in the range of 3 to 5. Since we think that (7.13), the dialogue generated from the comparable model, has lower naturalness score, its expected range of its naturalness score is between 1 and 3. From this aspect, we could examine the naturalness score of a dialogue against its expected range to know whether or not this dialogue caused the poor outcome of the evaluation.

Possible reasons for the poor outcome of testing the F1 function in the room arrangement domain

For the two dialogues for testing the F1 function in the room arrangement domain, the numbers of judgements falling out of the expected range are 3 out of 14 and 7 out of 14 respectively. We think the latter is the main cause of the poor result. The dialogue is shown in (7.15).

Sentence (a) in (7.15) contains the test point. The F1 mechanism is necessary to interpret the sentence correctly. This is because: the spatial relation "`right`" in (7.15 a) satisfies RULE 3.10 explained in Section §3.10 and is preferred to be the screen. Because some other words in the sentence, e.g., "`cabinet`" and "`wall`", clearly have the domain source, different sources appear in the components of the sentence. Obviously, this means that the F1 mechanism is needed.

However, (7.15) is assumed to be generated from the imaginary comparable model without the F1 mechanism, therefore, we arranged the dialogue to show the other

interpretation, which is also demonstrated in Figure 7.13. The dialogue and the figure shows that without the F1 mechanism the comparable model assigns the domain to be the source of the input words, including the word "`right`". Therefore, the intrinsic direction of the cabinet (see (7.15 b)) is used in the dialogue.

We thought that the interpretation was wrong, and anticipated that the subjects would notice the unnatural interpretation in (7.15), and would then give the dialogue a low naturalness value (i.e. less than 3). The pilot experiment fit in our anticipation quite well. Both subjects marked the dialogue with low naturalness values, and strongly preferred the screen source interpretation of the word "`right`" when they were interviewed. However, this is not the case for the rest of the experiment. In total, only six out of 14 subjects marked this dialogue as *unnatural* or *fair unnatural*, and two of them did so because they thought the word "`right`" was ambiguous rather than preferred the screen interpretation. Seven subjects gave *fair natural* or *natural*, and the remaining one marked it *so-so*.

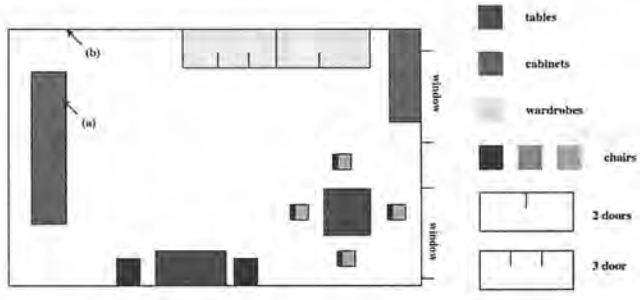


Figure 7.13: The screen display for (7.15) of the Room Arrangement domain

(7.15) **User:** What is the distance between this ↖(a) cabinet and the wall to its right? (a)
System: `You mean this` ↖(b) `wall? The distance is 48cm.` (b)

We think that there could be several explanations of the poor outcome, but we could not identify the actual one(s) without conducting further experiment.

The first one is that RULE 3.10, which could affect the appropriateness of using (7.15) to test the functionality of the F1 mechanism, could be invalid. That is, no favourite is given to screen relations over domain ones in a spatial domain. Therefore, the

judgement of some subjects might not actually test the F1 function, which results in the poor outcome we just mentioned.

Another possible explanation is that RULE 3.10 can still be assumed to be valid in general, but the rule can not be used appropriately here because no strong context indication restricts the screen interpretation of the word "`right`" to be the only right one. Therefore, some subjects might assume that what was provided in the dialogue was a possible interpretation, although they actually preferred a screen interpretation. We would have put more context in if the pilot experiment had not shown a strong bias towards the screen interpretation. In this case, further research is needed to find out which context information is needed in order to apply RULE 3.10.

A third possible explanation is that the system gave a confirmation in the dialogue (see the first clause of sentence (b) in (7.15)). The confirmation was added into the dialogue as a polish to clearly point out the wall so that the subjects had no difficulty in identifying the wall. Unfortunately, this confirmation might give some subjects the impression that the system was not sure about the user's input, and asked the user to confirm it, which is a natural action when there is potential misunderstanding. Therefore, the subjects might raise their verdicts on the naturalness of the dialogue even though they felt that there was misunderstanding.

Possible reasons for the poor outcome of testing the F4 function in the room arrangement domain

Following the same approach as above, we found that for the two dialogues for testing the F4 function in the room arrangement domain, 2 out of 14 and 4 out of 14 judgements fall out of the expected range respectively. The two results are rather similar, and we will examine both of them in the following.

(7.16) is a dialogue involving a model with the F4 function. Because the head noun "`icon`" of the phrase "`this` $\nwarrow_{(a)}$ `icon`" has the screen source, the model resolves the intended referent of the phrase to be a screen entity. Because the word "`right`" is the predicate that takes the intended referent of the phrase "`this` $\nwarrow_{(a)}$ `icon`" to be its argument if viewing from the perspective of QLE/LE expression, the source of the word "`right`" has to be the screen. 12 subjects agreed with this, and gave at least

a *so-so* (i.e. 3 to 5) to the naturalness of the dialogue. Only two subjects thought that the dialogue was unnatural. One of them believed that "`right`" should mean the "right of the cabinet" since the icon represented the cabinet. In other words, he preferred a domain interpretation. The other subject did so because she thought the meaning of "`right`" was ambiguous.

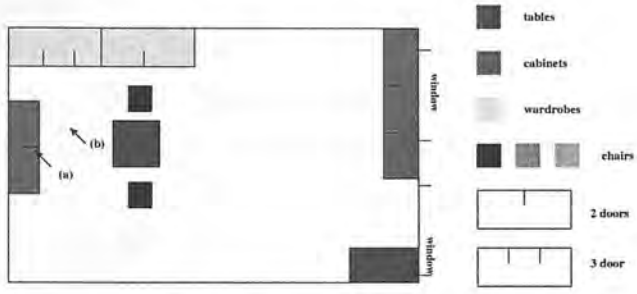


Figure 7.14: The screen display for (7.16) of the Room Arrangement domain

(7.16) **User:** How large is the gap at the right of this $\nwarrow_{(a)}$ icon? (a)
System: `You mean this` $\nwarrow_{(b)}$ `gap? It is 2.1cm.` (b)

(7.17) tests the F4 function in the room arrangement domain from a different perspective. We assume that the model does not have the F4 function, so for the purpose of contrast, the source of the word "`size`" was assigned to be the domain without considering the effect from the screen head noun "`icon`". This assignment can be inferred from the scale of the size provided by the model in (b) (i.e. 33cm x 66cm is a reasonable size for a table, not usually for an icon).

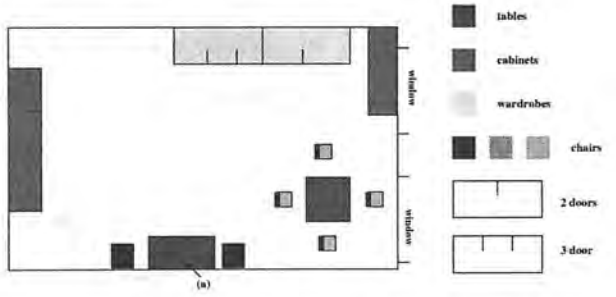


Figure 7.15: The screen display for (7.17) of the Room Arrangement domain

(7.17) **User:** What is the size of this $\nwarrow_{(a)}$ red icon? (a)
System: `Its size is 33cm x 66cm.` (b)

A natural expectation about the judgement of (7.17) would be that the majority of the subjects think the dialogue as unnatural. However, this is not the case.

Three subjects thought the dialogue was *so-so* (i.e. 3), and four subjects gave it even better naturalness scores. Among the remaining seven subjects who treated the dialogue as unnatural, only four thought that the unnatural part was the scale of the size generated by the model, which was due to the wrong source assignment. The other three believed that the word "`icon`" in (a) was misused, and preferred a word like "`table`" instead. This means that the three subjects did not question the correctness of the system answer, but rather that of the user input.

We do not know the exact reasons for some subjects to change their view dramatically between 7.16 and 7.17. One possibility is that "`size`" is a useful feature for real objects such as a piece of furniture, but not for an icon. People are probably more comfortable talking about the size of a table rather than that of an icon. Another possibility is the environment setting of the room arrangement domain, which might make the subjects think that all the conversations should be about furniture. This could be the reason behind the fact that three subjects wanted to change "`icon`" to "`table`".

The reason for the subjects to accept this dialogue could be that the icon represents a table and therefore it was not very strange for the system to return the size of the table.

The outcome from the dialogues testing the F4 function in the car selection domain was more consistent with our expectation. Only two subjects mentioned that they preferred to use "`car`" rather than "`icon`" in the dialogues.

Besides the above possible reasons, the difference between the two domains might also contribute to the dramatic differences in the test results. The major domain difference is that the spatial relations among icons in the car domain do not represent the real spatial relations among the represented entities, whereas those among the icons in the room domain do. This more closely bound relation between the icons and the

represented entities in the room domain might make the subjects ignore the existence of the icons more easily.

The conclusion based on the test results of the F4 function is that it works well most of the time, but we need to be more cautious when we use it in the room arrangement domain.

Further discussion

One thing often came up during the interview was that many subjects felt uncomfortable about the difference in colour between a domain object and its icon. However, most of them also said that they had no problem in understanding those dialogues.

Sometimes, the mixed use of the colours did confuse some subjects. One of them said that (7.4) was unnatural because all the cars on the screen were silver, so there was no green car on the screen at all. Not surprisingly, he gave the highest naturalness score to (7.5) for the same reason.

This indicates that although different colours can be used for domain entities and their icons, and sometimes it may be necessary to do so, this arrangement has to be used with caution because of the cognitive burden it may pose to the user.

Another interesting issue is the influence of pure screen operations on the context. In the experiment, the majority of the subjects thought that the context would be changed after pure screen operations, so the pronouns immediately after the operations would refer to the objects manipulated by the operations. However, one subject was clearly against this. He claimed that the operations would not change the context of the dialogues, so he gave the opposite naturalness scores as we anticipated to (7.7) and (7.8), the pair of dialogues for testing F5 in the car selection domain. Another subject felt that the pronouns after pure screen operations were ambiguous, but he did not have very strong opinions about them.

7.6 Summary

In this chapter, we discussed the evaluation of the resolution model, which aims at testing the usefulness of the model and revealing its design problems. To avoid the side effect from the limitations of the implementation, we used the overhearer method, a method that involves human subjects directly but the IMIG system indirectly.

Two dialogues were selected from the two domains, the car selection domain and the room arrangement domain. They were used to test the usefulness of the five distinctive functions of the resolution model. The dialogues and their screen displays were displayed on the screen and subjects were asked to make judgement about the naturalness of the dialogues and then mark the results on an answer sheet.

14 subjects were recruited, and all of their judgements were used in the statistical analyses. We used Wilcoxon Matched-Pairs Signed-Ranks Test on the collected data. The results confirm the usefulness of the model with high confidence.

However, some expected problems arose in the evaluation and they happened mostly in the room arrangement domain. We tried to provide some explanation about them.

Overall, the result of the evaluation shows that the model is useful.

Chapter 8

Conclusions

In this chapter, we summarise the contribution of this thesis, talk about future work and present some concluding remarks.

8.1 Contributions

The contributions of this thesis lie in the following aspects. Firstly, this research is about resolving source ambiguities appearing in multimodal interactions between a human and a computer where both natural language and graphic display are employed. Source ambiguities are problems arised when it is not clear whether the information of domain entities or of screen entities is used in an interaction. Although previous work briefly mentions some similar problems, it is our research that clarifies and identifies the source ambiguities and provides an infrastructure and a set of terminology for discussing them.

Secondly, through investigating many dialogues with source ambiguities, we have devised more than ten heuristics that are useful in our resolution model. These heuristics capture regularities in various constituents of a dialogue, ranging from referring expressions to sentences. They can also be used in other models for resolving source ambiguities.

Thirdly, to our knowledge, our work is the first study to represent visual display information and mapping relations in the system so that they can be used in the resolution process. We have designed a meaning representation language, MRL_S, which has the

ability of annotating information with different sources to facilitate localising computation in the resolution module.

Fourthly, our computational model uses a constraint satisfaction based method. This method enables the resolution process to flexibly access various knowledge from different knowledge bases (e.g. the display model, the world model, the mapping model, the general model and the context model). It is also capable of resolving source ambiguities and other referential ambiguities simultaneously.

Fifthly, the demonstration system we implemented has the computational model as its core and it integrates natural language input with graphic display. The system provides a concrete interface between users and the computer to fulfil certain tasks involving multimodal interactions.

Finally, the method we used for testing the ability of the model is the overhearer method. The method uses the subjects as the third parties who observe the dialogues between a user and the system. Through this, it is possible to involve the system indirectly so that only the aspects of the system related to the evaluation is revealed to the subjects, which helps to avoid the effect of implementation limitations of the system on the results of the evaluation.

8.2 Future work

Due to the large scale of the problems we are concerned with and the limitation in time, we had to make various simplifications in our research. Some of them can be interesting topics for future work.

8.2.1 Queries about mapping relations

In our current framework, mapping relations in the mapping model are accessible from the resolution model, but they would not be directly mentioned by the user in the dialogues. For example, the sentence "`which car is represented by a blue icon?`" is not in our consideration, although it is possible to be uttered in the interaction. This type of sentences imposes interesting problems to our current framework because

1. it directly interrogates the mapping relation, which, compared to the information in the domain and on the screen, is more like some kind of meta-knowledge of the system;

2. the source of the word "`represent`" is not the domain nor the screen since its relevant information is in the mapping model. If following our definition about source, the source of this word should be the mapping model.

Therefore, extensions to the framework (e.g. MRL_S, the CS resolution process, etc.) are needed in order to handle situations that there are three sources in the framework instead of two. However, we need further examination to find how the extensions can be done. This is why exploration on this aspect is an interesting future work.

8.2.2 Visual display knowledge in cross-media references

To handle source ambiguities, we constructed a knowledge management scheme that represents not only the domain knowledge, but also the knowledge about visual display and the mapping relations explicitly. This provides a clear structure to the resolution process for referring and accessing various types of knowledge. Although discussions about cross-media references in the literature have already mentioned using visual display attributes in the references, the ability of the system generating cross-media references using visual display attributes is limited because graphics are treated as descriptions only in their models (see Chapter 2). It is an interesting research work to apply our structure of organising knowledge to multimodal presentation tasks to extend the ability of the cross-media reference generator.

8.2.3 Multimodal dialogue corpus

Several regularities are used in the resolution process to help resolve source ambiguities. These regularities are mainly based on our intuition and experiments on a very small amount of test dialogues. This restricts the applicability of these regularities and imposes difficulty in developing the resolution model. It would be beneficial to have a multimodal dialogue corpus like those available for natural language text. Even just a

small corpus that only contains the dialogues related to our research would facilitate our further exploration on source ambiguities.

8.2.4 One anaphora in Multimodal context

"*One*" anaphora could be naturally used in complex referring expressions like "`all the blue cars to the right of the red one`". Although Dale talked about how to generate "*one*" anaphora [Dale, 1992], there is not enough work in interpreting "*one*" anaphora even in pure natural language text. It is an interesting work to exploring "*one*" anaphora in the context of multimodal interaction, especially in the situations that there could be source ambiguities. Because both the entities in the domain and those on the screen could be the one referred to by a referring expression, potentially more entities might have to be considered before an "*one*" anaphora could be resolved. However, more research is needed to identify the effect of a multimodal situation with source ambiguities to the resolution of "*one*" anaphora.

8.2.5 User intentions

Although the model we constructed can represent a small part of the cognitive state of the user by recognising whether or not an object has been mentioned in previous dialogues, any more complex function like modelling the user's intention is beyond the ability of the model.

When the dialogues are within the context of achieving a task, it might not be difficult to infer the mutual beliefs between the user (speaker) and the system (hearer) and the intentions behind the user's utterances. As with the linguistic regularities, the information about the beliefs and intentions can help the resolution process resolve source ambiguities because such information can select some particular entities from others. It is an interesting extension of the current model to include the help from the information about the beliefs and intentions.

8.2.6 Spoken language inputs

In the current system, the natural language input is assumed to be typed in through a keyboard. Speech input is an obvious choice for future extension on language input aspect. This extension can test the validity of the mechanisms we draw from written text when the language input is changed to spoken text. In addition, speech offers more friendly interaction between the computer and a wider range of end users.

8.2.7 Drawing and mapping relations

Although the screen displays in current application domains are dynamic where graphic entities can be added, moved or removed all the time, the mapping relations are usually static after the mapping relations have been decided. However, in some application domains, such as the situations that two people talk about a design of a room or a device by drawing on the screen, new entities/mapping relations can be introduced and old entities/mapping relations can be removed or modified all the time. It is interesting to think from the knowledge management aspect (i.e. organising and updating the mapping model, the display model and the world model) about issues like how to handle such situations.

8.3 Concluding remarks

Aiming at providing natural, effective and efficient communication, intelligent multimodal systems become increasingly important in human-computer interaction research. More and more systems have integrated multiple modalities, such as natural language, graphics, gestures and so on. However, demanding more powerful tools is always among the feedback from the communication with human users. We have described a resolution model that could enhance the multimodal system's reference processing ability and have proved that our approach is helpful through a set of experiments. Contributions of this thesis and future work have also presented to finish the whole picture of this work.

Bibliography

[Allen, 1993] Allen, J. (1993). Natural language, knowledge representation, and logical form. In Bates, M. and Weischedel, R. M., editors, *Challenges in natural language processing*, Studies in Natural Language Processing, chapter 6, pages 146–175. Cambridge University Press.

[Allen, 1995] Allen, J. (1995). *Natural Language Understanding*. Benjamin Cummings, Menlo Park, USA, second edition.

[Allgayer et al., 1989] Allgayer, J., Harbusch, K., Kobsa, A., Reddig, C., Reithinger, N., and Schmauks, D. (1989). XTRA: a natural language access system to expert systems. *International Journal of Man-Machine Studies*, 31:161–195.

[Alshawi, 1987] Alshawi, H. (1987). *Memory and Context for Language Interpretation*. Cambridge University Press, Cambridge UK.

[Alshawi, 1992] Alshawi, H., editor (1992). *The Core Language Engine*. MIT Press, Cambridge, MA.

[André et al., 1993] André, E., Finkler, W., Graf, W., Rist, T., Schauder, A., and Wahlster, W. (1993). WIP: The Automatic Synthesis of Multimedia Presentations. In Maybury, M., editor, *Intelligent Multimedia Interfaces*, pages 75–93. AAAI/MIT Press, Menlo Park, CA/Cambridge, MA. also in German Research Center for Artificial Intelligence(DFKI)'s research report, RR-92-46, 1992.

[André and Rist, 1993] André, E. and Rist, T. (1993). The Design of Illustrated Documents as a Planning Task. In Maybury, M., editor, *Intelligent Multimedia Interfaces*, pages 94–116. AAAI/MIT Press, Menlo Park, CA/Cambridge, MA.

[André and Rist, 1994] André, E. and Rist, T. (1994). Referring to World Objects with Text and Pictures. In *Proceedings of the 15th International Conference on Computational Linguistics(COLING'94)*, pages 530–534, Kyoto Japan.

[André and Rist, 1995] André, E. and Rist, T. (1995). Generating Coherent Presentations Employing Textual and Visual Material. *Artificial Intelligence Review*, 9:147–165.

[Androutsopoulos, 1992] Androutsopoulos, I. (1992). Interfacing a Natural Language Front-End to a Relational Database. Master's thesis, University of Edinburgh, Edinburgh, UK.

[Androutsopoulos, 1996] Androutsopoulos, I. (1996). *A Principled Framework for Constructing Natural Language Interfaces to Temporal Databases* . PhD thesis, University of Edinburgh, Edinburgh, UK.

[Antenucci et al., 1991] Antenucci, J., Brown, K., Croswell, P., Michael, K., and Archer, H. (1991). *Geographic Information Systems: A Guide to the Technology.* Van Nostrand Reinhold, New York, NY.

[Appelt, 1989] Appelt, D. E. (1989). Reference and Pragmatic Identification. In Wilks, Y., editor, *Theoretical Issues in Natural Language Processing*, pages 149–152. LEA, Hillsdale, NJ.

[Auxerre, 1986] Auxerre, P. (1986). MASQUE- Modular Answering System for Queries in English - Programmer's Manual. Technical report, AIAI, University of Edinburgh.

[Bateman et al., 1995] Bateman, J., Henschel, R., and Rinaldi, F. (1995). The generalized upper model 2.0. Technical report, GMD/IPSI, Darmstadt and University of Sydney.

[Beierle et al., 1990] Beierle, C., Pletat, U., and Studer, R. (1990). Knowledge Representation for Natural Language Understanding: The L_{LILOG} Approach. Technical Report IWBS Report 120, IBM Germany Science Center, Institute for Knowledge Based Systems, Stuttgart, Germany.

[Ben Amara et al., 1991] Ben Amara, H., Peroche, B., Chappel, H., and Wilson, M. (1991). Graphical interaction in a multimodal interface. In *Proceedings of Esprit Conferences*, pages 303–321. Kluwer Academic Publisher, Dordrecht, Netherlands.

[Bennett et al., 1996] Bennett, D., Armstrong, M., and Weirich, F. (1996). An Object-Oriented Model Base Management System for Environmental Simulation. In Goodchild, M., Steyaert, L., Parks, B., Johnston, C., Maidment, D., Michael, C., and Glendinning, S., editors, *GIS and Environmental Modelling:Progress and Research Issues*, pages 439–443. GIS World Inc., Fort Collins, CO.

[Bertin, 1983] Bertin, J. (1983). *Semiology of Graphics: Diagrams, Networks and Maps.* The University of Wisconsin Press, Madison, Wisconsin.

[Binot et al., 1992] Binot, J., Debille, L., Sedlock, D., Vandecapelle, B., Chappel, H., and Wilson, M. D. (1992). Multimodal integration in MMI^2: Anaphora resolution and mode selection. In Luczak, H., Cakir, A., and Cakir, G., editors, *Work With Display Units-WWDU'92 (Abstract Book of 3rd. International Scientific Conference)*, Berlin, Germany.

[Bos et al., 1994] Bos, E., Huls, C., and Claassen, W. (1994). EDWARD: full integration of language and action in a multimedia user interface. *International Journal of Human-Computer Studies*, 40:473–495.

[Brachman and Schmolze, 1985] Brachman, R. and Schmolze, J. (1985). An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9:171–216.

[Bunt, 1985] Bunt, H. (1985). *Mass terms and Model-theoretic semantics.* Cambridge University, Cambridge, UK.

[Bunt, 1998] Bunt, H. (1998). Issues in Multimodal Human-Computer Communication. In Bunt, H., Beun, R.-J., and Borghuis, T., editors, *Multimodal Human-Computer Communication: System, Techniques, and Experiments*, volume 1374 of *Lecture Notes in Artificial Intelligence*. Springer, Berlin Germany.

[Carenini et al., 1993] Carenini, G., Pianesi, F., Ponzi, M., and Stock, O. (1993). Natural-Language Generation and Hypertext Access. *Applied Artificial Intelligence*, 7(2):135–164.

[Carnap, 1964] Carnap, R. (1964). Foundations of logic and mathematics. In Fodor, J. and Katz, J., editors, *The structure of language:Reading in the Philosophy of Language*, pages 419–436. Prentice-Hall.

[Carpenter and Penn, 1994] Carpenter, B. and Penn, G. (1994). The Attribute Logic Engine, Version 2.0.1 User's Guide. Technical report, Computational Linguistics Program, Philosophy Department, Carnegie Mellon University.

[Carter, 1987] Carter, D. (1987). *Interpreting anaphors in natural language texts*. Ellis Horwood, Chichester, UK.

[Clark and Marshall, 1981] Clark, H. and Marshall, C. (1981). Definite reference and mutual knowledge. In Joshi, A., Webber, B., and Sag, I., editors, *Elements of discourse understanding*, chapter 1, pages 10–63. Cambridge University Press, Cambridge, UK.

[Cohen et al., 1989] Cohen, P., Dalrymple, M., Moran, D., Pereira, F., Sullivan, J., Gargan Jr, R., Schlossberg, J., and Tyler, S. (1989). Synergistic Use of Direct Manipulation and Natuarl Language. In *Proceedings of the 1989 Conference on Human Factors in Computing Systems (CHI'898)*, pages 227–233.

[Cox et al., 1999] Cox, R., McKendree, J., Tobin, R., Lee, J., and Mayes, T. (1999). Vicarious learning from dialogue and discourse: A controlled comparison. *Instructional Science*, 27:431–458.

[Dale, 1992] Dale, R. (1992). *Generating Referring Expressions*. A Bradford book, MIT Press, Cambridge, Massachusetts.

[Dale and Haddock, 1991] Dale, R. and Haddock, N. (1991). Content determination in the generation of referring expressions. *Computational Intelligence*, 7:252–265.

[De Smedt, 1987] De Smedt, K. (1987). Object-oriented programming in flavors and commonorbit. In Hawley, R., editor, *Artificial Intelligence Programming Environments*, pages 157–176. Ellis Horwood, Chichester.

[Dix, 1998] Dix, A. J. (1998). *Human-computer interaction*. Prentice Hall.

[Doe et al., 1992] Doe, G. J., Ringland, G. A., and Wilson, M. (1992). A meaning representation language for cooperative dialogue. In *Proceedings of the ERCIM Workshop on Experimental and Theoretical studies in Knowledge Representation*, pages 33–40, Pisa, Italy.

[Dowty et al., 1981] Dowty, D. R., Wall, R. E., and Peters, S. (1981). *Introduction to Montague semantics*, volume 11 of *Studies in Linguistics and Philosophy*. D. Reidel Publishing Company, Dordrecht, Holland.

[Ersan and Akman, 1994] Ersan, E. and Akman, V. (1994). Focusing for pronoun resolution in English discourse: an implementation. Technical Report BU-CEIS-94-29, Department of Computer Engineering and Information Science, Bilkent University.

[Feiner and McKeown, 1990a] Feiner, S. and McKeown, K. (1990a). Coordinating Text and Graphics in Explanation Generation. In *Proceedings of 8th National Conference of the American Association for Artificial Intelligence(AAAI'90)*, pages 442–449, Boston MA.

[Feiner and McKeown, 1990b] Feiner, S. and McKeown, K. (1990b). Generating Coordinated Multimedia Explanations. In *Proceedings the 6th Conference on Artificial Intelligence Application(CAIA-90)*, pages 290–296, Santa Barbara, CA.

[Feiner and McKeown, 1993] Feiner, S. and McKeown, K. (1993). Automating the Generation of Coordinated Multimedia Explanations. In Maybury, M., editor, *Intelligent Multimedia Interfaces*, pages 117–138. AAAI/MIT Press, Menlo Park, CA/Cambridge, MA.

[Fernandes et al., 1994] Fernandes, A. A., Ritchie, G., and Moffat, D. (1994). A Formal Reconstruction of Procedural Semantics. Technical Report DAI-RP-678, Department of Artificial Intelligence, University of Edinburgh.

[Franconi, 1990] Franconi, E. (1990). The YAK Manual: Yet another KRAPFEN. Technical Report Manual no. 9003-01, IRST, Trento, Italy.

[Frege, 1975] Frege, G. ([1892]1975). Sense and reference. In Davidson, D. and Harman, G., editors, *The Logic of Grammar*, pages 116–128. Dickenson, Encino, CA.

[Gale et al., 1992] Gale, W., Church, K. W., and Yarowsky, D. (1992). Estimating upper and lower bounds on the performance of word-sense disambiguation programs. In *Proceedings of ACL-92*.

[Gazdar and Mellish, 1989] Gazdar, G. and Mellish, C. (1989). *Natural Language Processing in PROLOG*. Addison-Wesley, Wokingham, UK.

[Grice, 1975] Grice, H. P. (1975). Logic and Conversation. In Cole, P. and Morgan, J. L., editors, *Syntax and Semantics*, volume 3:*Speech Acts*. Academic Press.

[Grosz, 1977] Grosz, B. (1977). The representation and Use of Focus in a System for Understanding dialogs. In *Proceedings of 5th International Joint Conference on Artificial Intelligence(IJCAI77)*.

[Grosz et al., 1995] Grosz, B., Joshi, A., and Weinstein, S. (1995). Centering: A Framework for Modeling the Local Coherence of Discourse. *Computational Linguistics*, 21(2):203–226.

[Grosz and Sidner, 1986] Grosz, B. and Sidner, C. (1986). Attention, Intentions, and the Structure of Discourse. *Computational Linguistics*, 12(3):175–204.

[Grover et al., 1993] Grover, C., Carroll, J., and Briscoe, T. (1993). The Alvey Natural Language Tools Grammar. Technical report, Human Communication Research Center, University of Edinburgh.

[Haddock, 1988] Haddock, N. J. (1988). *Incremental Semantics and Interactive Syntactic Processing.* PhD thesis, University of Edinburgh, Edinburgh, Scotland.

[Hajicova, 1987] Hajicova, E. (1987). Focusing: ameeting point of linguistics and artificial intelligence. In Jorand, P. and Sgurev, V., editors, *Artificial Intelligence II: Methodology, Systems, Applications.* Elsevier Science Publishers, Amsterdam, Netherlands.

[Halliday, 1994] Halliday, M. (1994). *An Introduction to Functional Grammar.* Edward Arnold, London, UK, second edition.

[Hatch and Lazaration, 1991] Hatch, E. M. and Lazaration, A. (1991). *The research manual: design and statistics for applied linguistics.* Newbury House, New York.

[Hawkins, 1978] Hawkins, J. A. (1978). *Definiteness and Indefiniteness: a study in reference and grammaticality prediction.* Croom Helm, London.

[Hayes, 1986] Hayes, P. J. (1986). Steps towards Integrating Natural Language and Graphical Interaction for Knowledge-based Systems. In *Proceedings of 7th European Conference on Artificial Intelligence(ECAI'86)*, pages 456–465, Brighton, UK.

[He et al., 1997] He, D., Ritchie, G., and Lee, J. (1997). Referring to Displays in Multimodal Interfaces. In *Referring Phenomena in a Multimedia Context and Their Computational Treatment, A workshop of ACL/EACL'97*, pages 79–82, Madrid, Spain.

[He et al., 1998] He, D., Ritchie, G., and Lee, J. (1998). Disambiguation between Visual Display and Represented Domain in Multimodal Interfaces. In *Combining AI and Graphics for the Interface of the Future, A workshop of ECAI'98*, pages 79–82, Brighton, UK.

[Hearst, 1991] Hearst, M. (1991). Toward Noun Homonym Disambiguation Using Local Context in Large Text Corpora. In *Proceedings of the Seventh Annual Conference of the UW Centre for the New OED*, pages 1–22, Oxford.

[Hobbs, 1978] Hobbs, J. R. (1978). Resolving pronoun references. *Lingua*, 44:311–338.

[Hobbs and Shieber, 1987] Hobbs, J. R. and Shieber, S. M. (1987). An algorithm for generating quantifier scopings. *Computational Linguistics*, 13(1-2):47–63.

[Huls et al., 1995] Huls, C., Bos, E., and Claassen, W. (1995). Automatic Referent Resolution of Deictic and Anaphoric Expressions. *Computational Linguistics*, 21(1):59–79.

[Johnston, 1998] Johnston, M. (1998). Unification-based multimodal parsing. In *Proceedings of the Conference of COLING-ACL'98*, Montreal, Canada.

[Kintsch, 1988] Kintsch, W. (1988). The role of knowledge in discourse comprehension: A construction integration model. *Psychological Review*, 95:163–182.

[Kobsa et al., 1986] Kobsa, A., Allgayer, J., Reddig, C., Reithinger, N., Schmauks, D., Harbusch, K., and Wahlster, W. (1986). Combining Deictic Gestures and Natural Language for Referent Identification. In *Proceedings of the 11th International Conference on Computational Linguistics(COLING'86)*, Bonn, Germany.

[Kronfeld, 1990] Kronfeld, A. (1990). *Reference and Computation: an essay in applied philosophy of language.* Studies in Natural Language Processing. Cambridge University Press, Cambridge, UK.

[Kumar, 1992] Kumar, V. (1992). Algorithms for constraint-satisfaction problems: A survey. *AI Magazine*, pages 32–44.

[Lee and Wang, 1996] Lee, C. and Wang, S.-C. (1996). Modelling GIS Data Using OMEGA. *Journal of Information Science and Engineering*, 12:345–364.

[Levelt, 1989] Levelt, W. J. M. (1989). *Speaking: From Intention to Articulation.* A Bradford Book, MIT Press, Cambridge, MA.

[Mackinlay, 1987] Mackinlay, J. (1987). Automating the design of graphical presentations of relational information. *Transactions on Graphics*, 5(2):110–141.

[Mackworth, 1977] Mackworth, A. K. (1977). Consistency in networks of relations. *Artificial Intelligence*, 8:99–118.

[Marslen-Wilson et al., 1982] Marslen-Wilson, W., Levy, E., and Tyler, L. K. (1982). Producing interpretable discourse: the establishment and maintenance of reference. In Jarvella, R. J. and Klein, W., editors, *Speech, Place and Action.* John Wiley and Sons Ltd., New York, NY.

[Maybury, 1993] Maybury, M. (1993). *Intelligent Multimedia Interfaces.* AAAI Press/MIT Press, Menlo Park, CA/Cambridge, MA.

[Maybury and Wahlster, 1998] Maybury, M. and Wahlster, W., editors (1998). *Readings in Intelligent user interfaces.* Morgan Kaufmann Publishers Inc., San Franciso, CA.

[McKeown et al., 1992] McKeown, K., Feiner, S., Robin, J., Seligmann, D., and Tanenblatt, M. (1992). Generating Cross-References for Multimedia Explanation. In *Proceedings of 10th National Conference of the American Association for Artificial Intelligence(AAAI'92)*, pages 9–16, San Jose CA.

[McRoy, 1992] McRoy, S. (1992). Using Multiple Knowledge Sources for Word Sense Discrimination. *Computational Linguistics*, 18(1).

[Mellish, 1985] Mellish, C. S. (1985). *Computer interpretation of natural language descriptions.* Ellis Horwood series in artificial intelligence. Ellis Horwood.

[Nadel, 1985] Nadel, B. (1985). The Consistent Labelling Problem, Part 1: Background and Problem Formulation. Technical Report CRL-TR-13-85, Computing Research Laboratory, University of Michigam.

[Neal et al., 1988] Neal, J., Dobes, Z., Bettinger, K., and Byoun, J. (1988). Multimodal References in Human-Computer Dialogue. In *Proceedings of 6th National Conference of the American Association for Artificial Intelligence(AAAI'88)*, pages 819–823, St. Paul MN.

[Neal and Shapiro, 1991] Neal, J. and Shapiro, S. (1991). Intelligent Multi-media Interface Technology. In Sullivan, J. and Tyler, S., editors, *Intelligent User Interfaces*, pages 11–44. ACM Press, New York, NY.

[Ousterhout, 1990] Ousterhout, J. K. (1990). Tcl: An Embeddable Command Language. In *Proceedings of 1990 Winter USENIX conference*, pages 133–146.

[Ousterhout, 1991] Ousterhout, J. K. (1991). An X11 Toolkit Based on the Tcl Language. In *Proceedings of 1991 Winter USENIX conference*, pages 133–146.

[Oviatt et al., 1997] Oviatt, S., DeAngeli, A., and Kuhn, K. (1997). Integration and Synchronization of Input Modes during Multimodal Human-Computer Interaction. In *Referring Phenomena in a Multimedia Context and Their Computational Treatment, A Meeting of the ACL Spefial Interest Group on Multimedia Language Processing*, pages 1–13, Madrid, Spain.

[Partee and Hendriks, 1997] Partee, B. H. and Hendriks, H. L. W. (1997). *Montague Grammar*, chapter 1, pages 5–91. Elsevier Science, Amsterdam Holland.

[Pollard and Sag, 1994] Pollard, C. and Sag, I. (1994). *Head-Driven Phrase Structure Grammar*. The University of Chicago Press, Chicago.

[Quirk et al., 1985] Quirk, R., Greenbaum, S., Leech, G., and Svartvik, J. (1985). *A comprehensive grammar of the english language*. Longman, London, UK.

[Rich and Knight, 1991] Rich, E. and Knight, K. (1991). *Artificial Intelligence*. McGraw-Hill, New York, 2 edition.

[Rossi et al., 1989] Rossi, F., Petrie, C., and Dhar, V. (1989). Equivalence of constraint-satisfaction problems. Technical Report ACT-AI-222-89, MCC Corp., Austin, Texas.

[Russell, 1905] Russell, B. (1905). On denoting. *Mind*, 14:479–493. Reprinted in 1985, *Logic and Knowledge*, ed. R.C. Marsh. London:George Allen and Unwin.

[Russell, 1953] Russell, B. ((1910)1953). Knowledge by acquaintance and knowledge by description. In *Mysticism and Logic*, chapter 10, pages 152–167. Penguin, Harmondsworth, UK.

[Russell and Norvig, 1995] Russell, S. and Norvig, P. (1995). *Artificial Intelligence: A Modern Approach*. Prentice Hall, Englewood Cliffs, NJ USA.

[Samek-Lodovici and Strapparava, 1990] Samek-Lodovici, V. and Strapparava, C. (1990). Identifying Noun Phrase References: The Topic Module of the ALFresco System. In *Proceedings of 9th European Conference on Artificial Intelligence(ECAI'90)*, Sweden.

[Schmauks, 1987] Schmauks, D. (1987). Natural and simulated pointing. In *Proceedings of 3rd Conference of the European Chapter of Association for Computational Linguistics*, pages 179–185, Copenhagen, Denmark.

[Schmauks and Reithinger, 1988] Schmauks, D. and Reithinger, N. (1988). Generating Multimodal Output - Conditions, Advantages and Problems. In *Proceedings of the 12th International Conference on Computational Linguistics (COLING'88)*, pages 584–588, Budapest, Hungary.

[Seldrup, 1995] Seldrup, I. (1995). A Customisable HPSG-based Natural Language Front-End. Master's thesis, Department of Artificial Intelligence, University of Edinburgh, Edinburgh, UK.

[SICStus, 1996] SICStus (1996). *SICStus Prolog User's Manual.* The Intelligent Systems Laboratory, Swedish Institute of Computer Science, Kista Sweden, 3.5 edition.

[Sidner, 1987] Sidner, C. (1987). Focusing in the Comprehension of Definite Anaphora. In Grosz, B., Jones, K. S., and Webber, B. L., editors, *Reading in Natural Language Processing*, pages 363–394. Morgan Kaufmann, Los Altos, CA.

[Stock and Team, 1993] Stock, O. and Team, A. (1993). ALFresco: Enjoying the Combination of NLP and Hypermedia for Information Exploration. In Maybury, M., editor, *Intelligent Multimedia Interfaces*, pages 197–224. AAAI/MIT Press, Menlo Park, CA/Cambridge, MA.

[Tang et al., 1996] Tang, A. Y., Adams, T. M., and Usery, E. L. (1996). A spatial data model design for feature-based geographical information systems. *International Journal of Geographical Information Systems*, 10(5):643–659.

[USENIX, 1998] USENIX (1998). *Proceedings of the 6th Annual USENIX Tcl/Tk Conference 1998.*

[Vieira, 1997] Vieira, R. (1997). *Definite Description Processing in Unrestricted Text.* PhD thesis, University of Edinburgh, Edinburgh, Scotland.

[Vieira and Poesio, 1997] Vieira, R. and Poesio, M. (1997). Processing definite descriptions in corpora. In Botley and McEnery, editors, *Corpus-based and computational approaches to anaphora.* forthcoming.

[Wahlster, 1991] Wahlster, W. (1991). User and Discourse Models for Multimodal Communication. In Sullivan, J. and Tyler, S., editors, *Intelligent User Interfaces*, pages 45–67. ACM Press, New York, NY.

[Wahlster et al., 1991] Wahlster, W., Andre, E., Graf, W., and Rist, T. (1991). Designing Illustrated Texts: How Language Production is Influenced by Graphics Generation. Technical Report RR-91-05, DFKI, Germany.

[Walker, 1997] Walker, M. (1997). Centering, anaphora resolution, and discourse structure. In Walker, M., Joshi, A., and Prince, E., editors, *Centering Theory in Discourse.* Clarendon Press, Oxford, England.

[Weischedel, 1989] Weischedel, R. (1989). A hybrid approach to representation in the JANUS natural language processor. In *Proceedings of the 27th Annual Meeting of the Association of Computational Linguistics*, pages 193–202.

[Wilson et al., 1991] Wilson, M. D., Sedlock, D., Binot, J.-L., and Falzon, P. (1991). An architecture for multimodal dialogue. In *Venaco 2nd Multi-Modal Workshop'91*.

[Woods, 1967] Woods, W. (1967). *Semantics for a Question-Answering System*. PhD thesis, Harvard University.

[Woods, 1968] Woods, W. (1968). Procedural semantics for a question-answering machine. In *the Proceedings for the Fall Joint Computer Conference*, pages 457–471, New York, NY. AFIPS.

[Woods, 1987] Woods, W. A. (1987). Natural language question answering. In Grosz, B., Jones, K. S., and Webber, B. L., editors, *Reading in Natural Language Processing*, pages 205–248. Morgan Kaufmann.

[Worboys, 1994] Worboys, M. F. (1994). Object-oriented approaches to geo-referenced information. *International Journal of Geographical Information Systems*, 8(4):385–399.

[Zancanaro et al., 1997] Zancanaro, M., Stock, O., and Strapparava, C. (1997). Multimodal interaction for information access: exploiting cohesion. *Computational Intelligence*, 13(4).

Appendix A

BNF descriptions of the MRL_S

A.1 BNF-like description for the current LE

The meta-notation used in the BNF description of the MRL_S are:

- The sign "|" denotes alternative RHSs, that is, a production rule which says:

  designator $\longrightarrow$ constant
  | variable name
  | function

  is equivalent to the set of production rules:

  designator $\longrightarrow$ constant
  designator $\longrightarrow$ variable name
  designator $\longrightarrow$ function

- The sign "$\{...\}^{\#}$" denotes a non-empty sequence of indefinite extension of instances of the symbol appended by it, i.e. $\{$`,formula`$\}^{\#}$ is equivalent to `, formula, formula ...`

- The distinguished symbol appears in SMALL CAPITALS. Here it is LOGICAL EXPRESSION;

- The non-terminals are written in plain texts;

- The terminal symbols are written in `Typewriter`.

LOGICAL EXPRESSION	$\longrightarrow$	command
command	$\longrightarrow$	`[test,` proposition`]` |
		`[list,` designator`]` |
		`[act,` action`]`
proposition	$\longrightarrow$	atomic proposition |
		complex proposition |
		quantified proposition
atomic proposition	$\longrightarrow$	[predicate symbol{, designator}$^{\#}$]
quantified proposition	$\longrightarrow$	`quant(`quantifier, variable, proposition, proposition`)`
complex proposition	$\longrightarrow$	[logic operator{, proposition}$^{\#}$]
predicate symbol	$\longrightarrow$	`pred(`predicate name, source`)`
action	$\longrightarrow$	`[oper(`action name, source`)`{, designator}$^{\#}$`]`
designator	$\longrightarrow$	constant |
		variable |
		`[map(corresobj,` source`)`{, designator}$^{\#}$`]` |
		quantified designator |
		[logic operator, designator]
quantified designator	$\longrightarrow$	`quant(interrog,` variable, proposition, proposition`)`
constant	$\longrightarrow$	`cons(`constant name, source`)`
variable	$\longrightarrow$	`var(`variable name, source`)`
constant name	$\longrightarrow$	individual name |
		number |
		pair |
predicate name	$\longrightarrow$	`is_blue` | `is_small` | `is_car` |...
variable name	$\longrightarrow$	`X1` | `Y1` | ...
action name	$\longrightarrow$	`move` | `delete` | `add`
logic operator	$\longrightarrow$	`not` | `and` | `or`
quantifier	$\longrightarrow$	`forall` | `exists` | `interrog`
source	$\longrightarrow$	`domain` | `screen`
individual name	$\longrightarrow$	`car1` | `nissan1` |...
pair	$\longrightarrow$	`pair(` integer, integer`)`
number	$\longrightarrow$	integer | real

Table A.1: BNF of LE

A.2 BNF description of QLE

QLE is a superset of LE. All the above BNF definitions for LE belong to QLE, except the first line, which is about LOGICAL EXPRESSION. As there is no need to rewrite the identical descriptions again, only those QLE definitions that do not appear in the definitions of LE are presented below. The notation rules are the same as those for LE.

QLE	$\longrightarrow$	command
proposition	$\longrightarrow$	quasi pred
quasi pred	$\longrightarrow$	`[qpred(pred name) , designator]`
designator	$\longrightarrow$	`qterm(category, variable, proposition)`
variable	$\longrightarrow$	`qvar(variable name)`
category	$\longrightarrow$	`[typ:`value$_1$`, phr:` value$_2$`, lex:` value$_3$ `, num:` value$_4$`, quant:`value$_5$`]`
value$_1$	$\longrightarrow$	`ref` | `quant`
value$_2$	$\longrightarrow$	`ana` | `def` | `dex`
value$_2$	$\longrightarrow$	string
value$_4$	$\longrightarrow$	`sing` | `plur`
value$_5$	$\longrightarrow$	`exists` | `forall` | `interrog`

Table A.2: BNF of QLE

Appendix B

Some Selected Dialogues

In this section, we are going to enumerate the 20 dialogues we selected for the evaluation. There are two versions presented in below for each selected dialogue. The first one is the original one we used in the interaction with the IMIG system and does not contain any polish, whereas the second one is the post-polish version, which was used in the evaluation. There is only one graphic display accompanying with each dialogue since it does not involved in polishing.

B.1 Episode 1

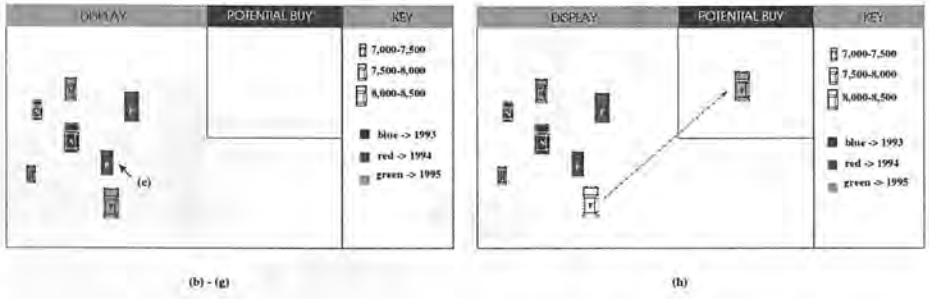


Figure B.1: The screen display for Episode 1 of the Car Selection System

The original dialogue

(B.1)
User: Is there a silver car? (a)
System: `Yes. All of them are displayed on the screen.` (b)
User: Is this $\nwarrow_{(c)}$ car a five-door hatchback? (c)
System: `Yes, it is.` (d)
User: Is the green car near it a five-door hatchback? (e)
System: `No, it is a three-door hatchback.` (f)
User: Move the green car to the potential buy area. (g)
System: `It has been moved.` (h)

The polished dialogue

(B.2)
User: Is there a silver car in your stock? (a)
System: `Yes, we have quite a few. Here all the silver cars are displayed on the screen.` (b)
User: Is this $\nwarrow_{(c)}$ car a five-door hatchback? (c)
System: `Yes, it is.` (d)
User: What about the green car near it then? (e)
System: `No, it is a three-door hatchback.` (f)
User: Move the green car to the potential buy area. (g)
System: `The green car has been moved.` (h)

B.2 Episode 2

Figure B.2: The screen display for Episode 2 of the Room Arrangement System

The original dialogue

(B.3) **User:** How large is the gap at the left of this $\nwarrow_{(a)}$ cabinet? (a)
System: `It is 18cm.` (b)

The polished dialogue

(B.4) **User:** How large is the gap at the left of this $\nwarrow_{(a)}$ cabinet? (a)
System: `You mean this` $\nwarrow_{(b)}$ `gap? It is 18cm.` (b)

B.3 Episode 3

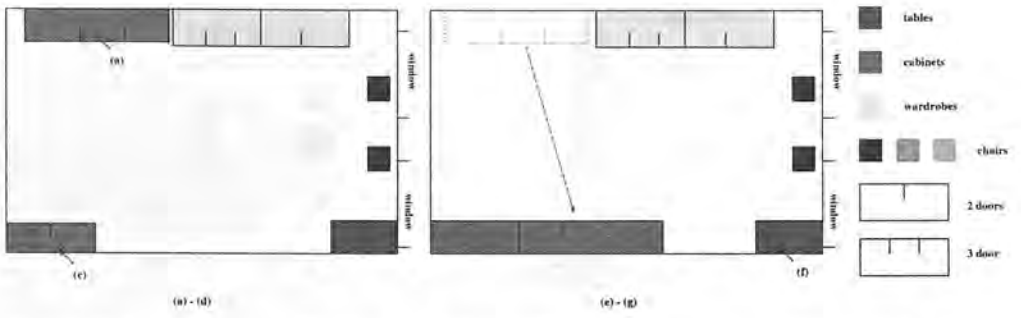


Figure B.3: The screen display for Episode B.3 of the Room Arrangement System

The original dialogue

(B.5)
User: What is the size of this $\nwarrow_{(a)}$ long cabinet? (a)
System: `It is 33cm x 142cm.` (b)
User: What is the size of this $\nwarrow_{(c)}$ short cabinet? (c)
System: `It is 33cm x 86cm.` (d)
Action: *The user moves the long cabinet to the right end of and alongside the shortcabinet (see Figure B.3 (e) - (g)).* (e)
User: What is the distance between the cabinet and this $\nwarrow_{(f)}$ table? (f)
System: `It is 93 cm.` (g)

The polished dialogue

(B.6)
User: What is the size of this $\nwarrow_{(a)}$ long cabinet? (a)
System: `33cm x 142cm.` (b)
User: How about this $\nwarrow_{(c)}$ short cabinet? (c)
System: `33cm x 86cm.` (d)
Action: *The user moves the long cabinet to the right end of and alongside the short cabinet (see Figure B.3 (e) - (g)).* (e)
User: What is the distance between the cabinet and this $\nwarrow_{(f)}$ table? (f)
System: `The distance between the long cabinet and the table is 93 cm.` (g)

B.4 Episode 4

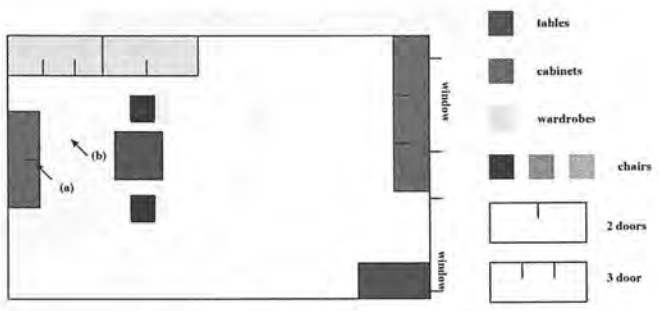


Figure B.4: The screen display for Episode 4 of the Room Arrangement System

The original dialogue

(B.7) **User:** How large is the gap at the right of this $\nwarrow_{(a)}$ icon? (a)
System: `It is 2.1cm.` (b)

The polished dialogue

(B.8) **User:** How large is the gap at the right of this $\nwarrow_{(a)}$ icon? (a)
System: `You mean this` $\nwarrow_{(b)}$ `gap? It is 2.1cm.` (b)

B.5 Episode 5

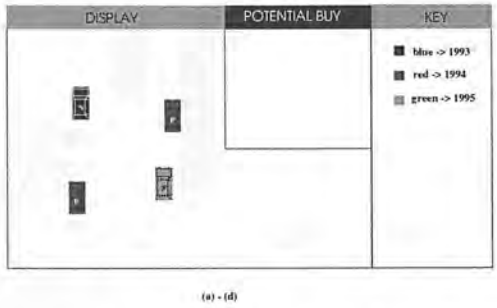


Figure B.5: The screen display for Episode 5 of the Car Selection System

The original dialogue

(B.9)
User: How much is the green car? (a)
System: `It is 14,200 pounds.` (b)
User: Remove it from the screen. (c)
System: `This operation can only be used on icons.` (d)

The polished dialogue

(B.10)
User: How much is the green car? (a)
System: `14,200 pounds.` (b)
User: It is too expensive. Remove it from the screen. (c)
System: `Remove can only be used on icons, but your phrase ''it'' refers to a car.` (d)

B.6 Episode 6

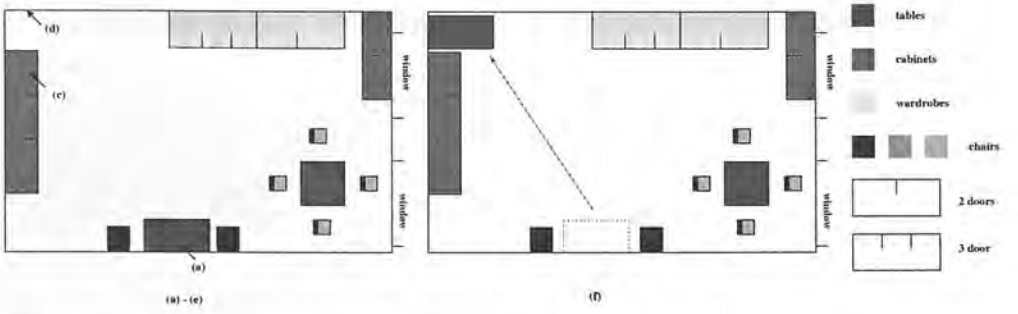


Figure B.6: The screen display for Episode 6 of the Room Arrangement System

The original dialogue

(B.11) **User:** What is the size of this $\nwarrow_{(a)}$ table? (a)
System: `It is 33cm x 66cm.` (b)
User: What is the distance between this $\nwarrow_{(c)}$ cabinet and the wall above it? (c)
System: `It is 48cm.` (d)
User: Move the table to the place above the cabinet. (e)
System: `It has been moved.` (f)

The polished dialogue

(B.12) **User:** What is the size of this $\nwarrow_{(a)}$ table? (a)
System: `33cm x 66cm.` (b)
User: What is the distance between this $\nwarrow_{(c)}$ cabinet and the wall above it? (c)
System: `You mean this` $\nwarrow_{(d)}$ `wall? The distance is 48cm.` (d)
User: Move the table to the place above the cabinet. (e)
System: `The table has been moved.` (f)

B.7 Episode 7

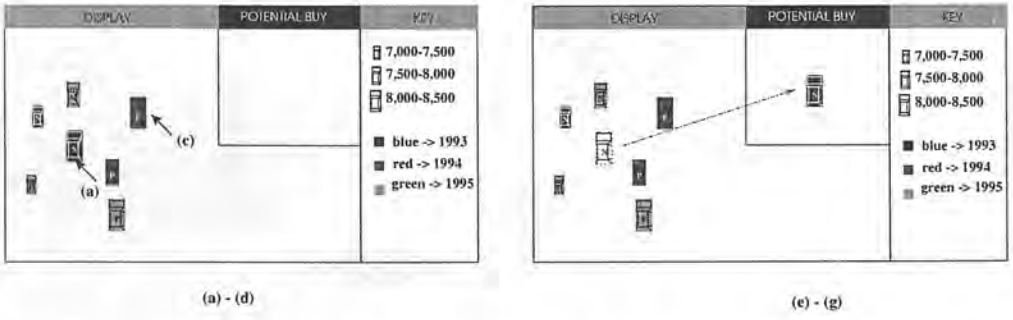


Figure B.7: The screen display for Episode 7 of the Car Selection System

The original dialogue

(B.13)
User: What is the insurance group of this $\nwarrow_{(a)}$ car? (a)
System: `It is in group 5.` (b)
User: What is the insurance group of this $\nwarrow_{(c)}$ red car? (c)
System: `It is in group 8.` (d)
Action: *The user moves the icon of the car mentioned in (a) to the potential buy area (see Figure B.7 (e) - (g)).* (e)
User: Give me all information about it. (f)
System: `It is ...` (g)

The polished dialogue

(B.14)
User: What is the insurance group of this $\nwarrow_{(a)}$ car? (a)
System: `It is in group 5.` (b)
User: What is the insurance group of this $\nwarrow_{(c)}$ red car then? (c)
System: `It is in group 8.` (d)
Action: *The user moves the icon of the car mentioned in (a) to the potential buy area (see Figure B.7 (e) - (g)).* (e)
User: Give me more information about it. (f)
System: `You already know that its insurance group is 5. The other information is ...` (g)

B.8 Episode 8

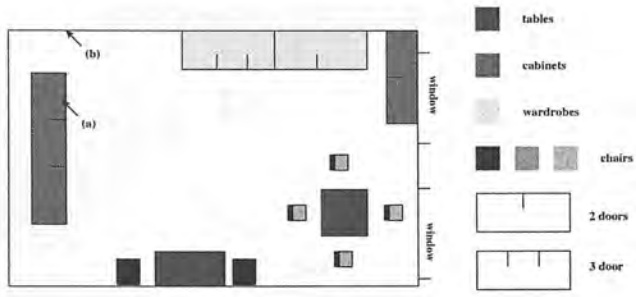


Figure B.8: The screen display for Episode 8 of the Room Arrangement System

The original dialogue

(B.15) **User:** What is the distance between this $\nwarrow_{(a)}$ cabinet and the wall to its right? (a)
System: `It is 48cm.` (b)

The polished dialogue

(B.16) **User:** What is the distance between this $\nwarrow_{(a)}$ cabinet and the wall to its right? (a)
System: `You mean this` $\nwarrow_{(b)}$ `wall? The distance is 48cm.` (b)

B.9 Episode 9

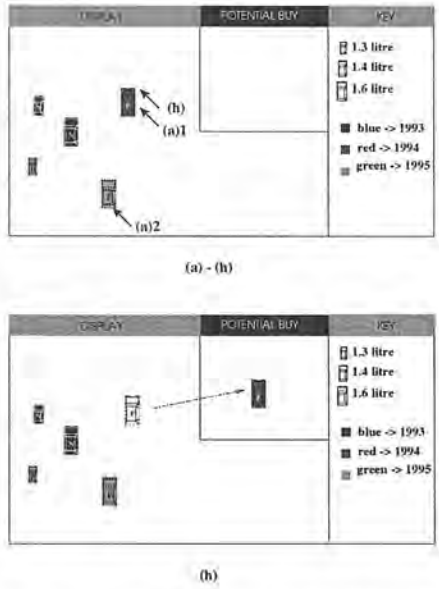


Figure B.9: The screen display for Episode 9 of the Car Selection System

The original dialogue

(B.17)
User: What is the colour of this $\nwarrow_{(a)1}$ car and this $\nwarrow_{(a)2}$ car? (a)
System: `The first car is green, and the second one is white.` (b)
User: Is the green car a five-door hatchback? (c)
System: `Yes, it is.` (d)
User: Is the white car a five-door hatchback? (e)
System: `No, it is not. It is a saloon.` (f)
User: Move the green icon to the potential buy area. (g)
System: *[Action: The icon is moved.]* `It has been moved.` (h)

The polished dialogue

(B.18)
User: What is the colour of this $\nwarrow_{(a)1}$ car and this $\nwarrow_{(a)2}$ car? (a)
System: `The first car is green, and the second one is white.` (b)
User: Is the green car a five-door hatchback? (c)
System: `Yes, it is.` (d)
User: How about the white car? (e)
System: `No, it is not. It is a saloon.` (f)
User: Move the green icon to the potential buy area. (g)
System: `You mean this` $\nwarrow_{(h)}$ `icon?` *[Action: The icon is moved.]*
`It has been moved.` (h)

B.10 Episode 10

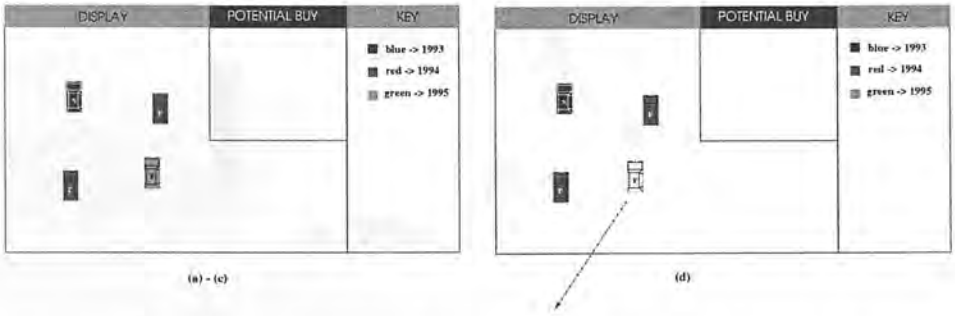


Figure B.10: The screen display for Episode 10 of the Car Selection System

The original dialogue

(B.19) **User:** What is the price of the green car? (a)
System: `It is 14,200 pounds.` (b)
User: Remove it from the screen. (c)
System: `It has been removed.` (d)

The polished dialogue

(B.20) **User:** How much is the green car? (a)
System: `14,200 pounds.` (b)
User: It is too expensive. Remove it from the screen. (c)
System: `The green car has been removed from the screen.` (d)

B.11 Episode 11

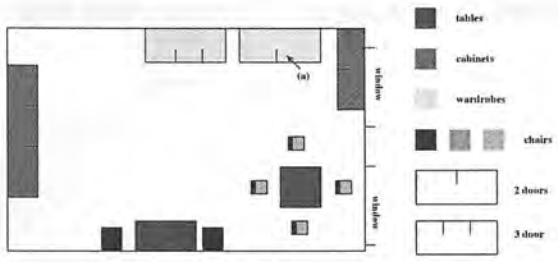


Figure B.11: The screen display for Episode 11 of the Room Arrangement System

The original dialogue

(B.21)
User: What is the size of this $\nwarrow_{(a)}$ wardrobe? (a)
System: `It is 36cm x 72cm.` (b)
User: What is the size of the cabinet near that icon? (c)
System: `Error: I can not find the icon.` (d)

The polished dialogue

(B.22)
User: What is the size of this $\nwarrow_{(a)}$ wardrobe? (a)
System: `36cm x 72cm.` (b)
User: And the cabinet near that icon? (c)
System: `Near which icon? I do not know.` (d)

B.12 Episode 12

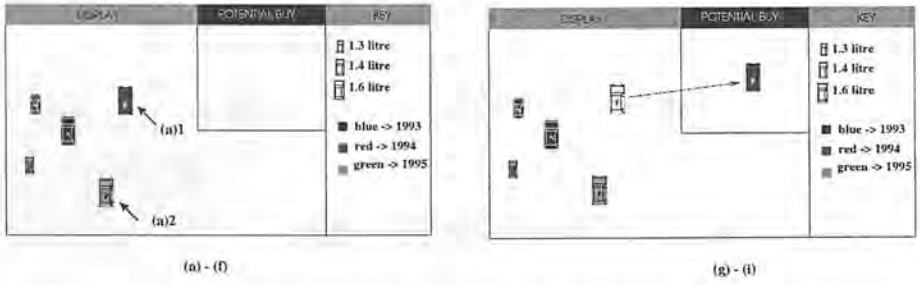


Figure B.12: The screen display for Episode 12 of the Car Selection System

The original dialogue

(B.23) **User:** What is the colour of this $\nwarrow_{(a)1}$ car and this $\nwarrow_{(a)2}$ car? (a)
System: `The first car is green, and the second one is white.` (b)
User: What is the price of the green car? (c)
System: `It is 14,200 pounds.` (d)
User: What is the price of the white car? (e)
System: `It is 13,000 pounds.` (f)
Action: *After reading the answer of the system, the user moves the icon of the green car to the potential buy area (see Figure B.12 (g) - (i)).* (g)
User: Give me all information about it. (h)
System: `It is ...` (i)

The polished dialogue

(B.24) **User:** What is the colour of this $\nwarrow_{(a)1}$ car and this $\nwarrow_{(a)2}$ car? (a)
System: `The first car is green, and the second one is white.` (b)
User: How much is the green car? (c)
System: `14,200 pounds.` (d)
User: How about the white car? (e)
System: `13,000 pounds.` (f)
Action: *After reading the answer of the system, the user moves the icon of the green car to the potential buy area (see Figure B.12 (g) - (i)).* (g)
User: Give me more information about it. (h)
System: `You already know that the price of this car is 14,200 pounds and it is a green car. The other information is ...` (i)

B.13 Episode 13

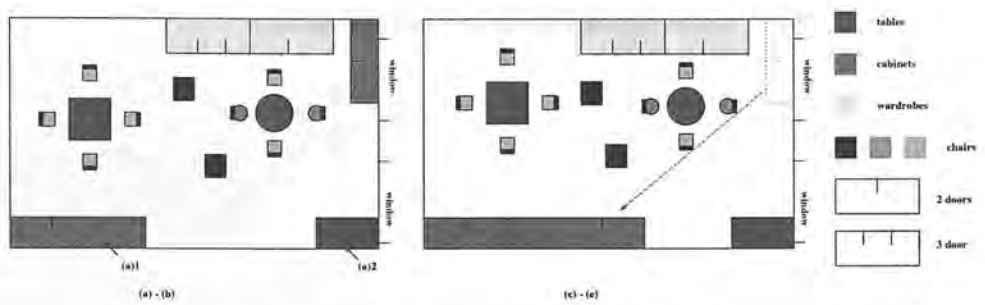


Figure B.13: The screen display for Episode 13 of the Room Arrangement System

The original dialogue

(B.25) **User:** What is the distance between this $\nwarrow_{(a)1}$ long cabinet and this $\nwarrow_{(a)2}$ long table? (a)
System: `It is 179 cm.` (b)
Action: *The user moves the icon of the short cabinet to the place alongside the long cabinet (see Figure B.13 (c)-(e)).* (c)
User: What is the distance between the cabinet and the long table? (d)
System: `It is still 179 cm.` (e)

The polished dialogue

(B.26) **User:** What is the distance between this $\nwarrow_{(a)1}$ long cabinet and this $\nwarrow_{(a)2}$ long table? (a)
System: `The distance is 179 cm.` (b)
Action: *The user moves the icon of the short cabinet to the place alongside the long cabinet (see Figure B.13 (c)-(e)).* (c)
User: What is the distance between the cabinet and the long table? (d)
System: `The distance between the long cabinet and the long table is still 179 cm.` (e)

B.14 Episode 14

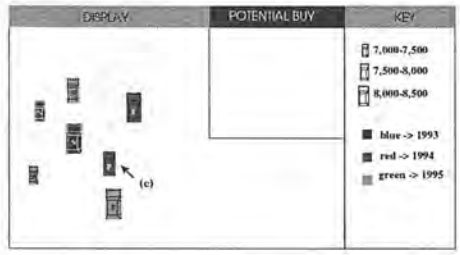


Figure B.14: The screen display for Episode 14 of the Car Selection System

The original dialogue

(B.27)
User: Is there a silver car? (a)
System: `Yes. All of them are displayed on the screen.` (b)
User: Is this $\nwarrow_{(c)}$ car a five-door hatchback? (c)
System: `Yes, it is.` (d)
User: Is the green car near it a five-door hatchback? (e)
System: `There is no green car.` (f)

The polished dialogue

(B.28)
User: Is there a silver car in your stock? (a)
System: `Yes, we have quite a few. Here, all the silver cars are displayed on the screen for you.` (b)
User: Is this $\nwarrow_{(c)}$ car a five-door hatchback? (c)
System: `Yes, it is.` (d)
User: What about the green car near it then? (e)
System: `In our stock, there is no green car near that five-door hatchback.` (f)

B.15 Episode 15

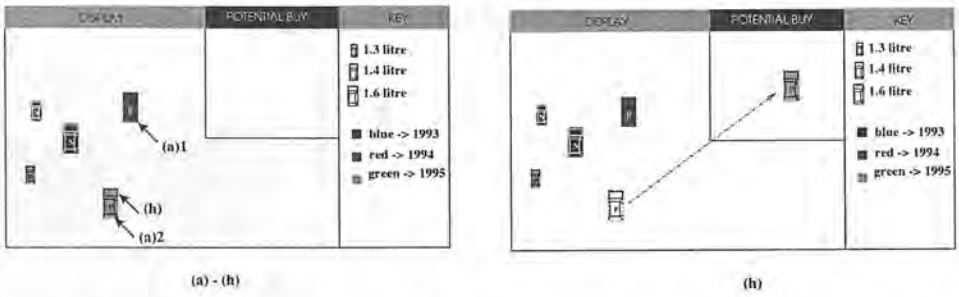


Figure B.15: The screen display for Episode 15 of the Car Selection System

The original dialogue

(B.29) **User:** What is the colour of this $\nwarrow_{(a)1}$ car and this $\nwarrow_{(a)2}$ car? (a)
System: `The first car is green, and the second one is white.` (b)
User: Is the green car a five-door hatchback? (c)
System: `Yes, it is.` (d)
User: Is the white car a five-door hatchback? (e)
System: `No, it is not. It is a saloon.` (f)
User: Move the green icon to the potential buy area. (g)
System: *[Action: The icon is moved.]* `It has been moved.` (h)

The polished dialogue

(B.30) **User:** What is the colour of this $\nwarrow_{(a)1}$ car and this $\nwarrow_{(a)2}$ car? (a)
System: `The first car is green, and the second one is white.` (b)
User: Is the green car a five-door hatchback? (c)
System: `Yes, it is.` (d)
User: How about the white car? (e)
System: `No, it is not. It is a saloon.` (f)
User: Move the green icon to the potential buy area. (g)
System: `You mean this` $\nwarrow_{(h)}$ `icon?` *[Action: The icon is moved.]* `It has been moved.` (h)

B.16 Episode 16

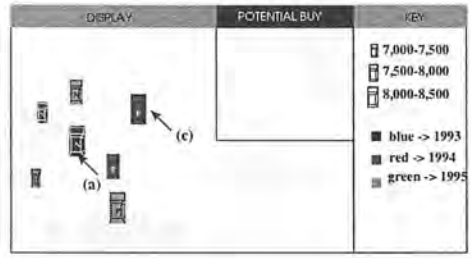


Figure B.16: The screen display for Episode 16 of the Car Selection System

The original dialogue

(B.31)
User: What is the insurance group of this $\nwarrow_{(a)}$ car? (a)
System: `It is in group 5.` (b)
User: What is the insurance group of this $\nwarrow_{(c)}$ red car? (c)
System: `It is in group 8.` (d)
User: Give me all information about it. (e)
System: `It is ...` (f)

The polished dialogue

(B.32)
User: What is the insurance group of this $\nwarrow_{(a)}$ car? (a)
System: `It is in group 5.` (b)
User: What is the insurance group of this $\nwarrow_{(c)}$ red car then? (c)
System: `It is in group 8.` (d)
User: Give me more information about it. (e)
System: `You already know that its insurance group is 5. The other information is ...` (f)

B.17 Episode 17

Figure B.17: The screen display for Episode 17 of the Room Arrangement System

The original dialogue

(B.33) **User:** What is the size of this $\nwarrow_{(a)}$ table? (a)
System: `53cm x 79cm.` (b)
User: What is the distance between this $\nwarrow_{(c)}$ cabinet and the wall above it? (c)
System: `It is 60cm.` (d)
User: Move the table to the place above the cabinet. (e)
System: `It has been moved.` (f)

The polished dialogue

(B.34) **User:** What is the size of this $\nwarrow_{(a)}$ table? (a)
System: `53cm x 79cm.` (b)
User: What is the distance between this $\nwarrow_{(c)}$ cabinet and the wall above it? (c)
System: `You mean this` $\nwarrow_{(d)}$ `wall? The distance is 60cm.` (d)
User: Move the table to the place above the cabinet. (e)
System: `You mean like this` $\nwarrow_{(f)}$`?` (f)

B.18 Episode 18

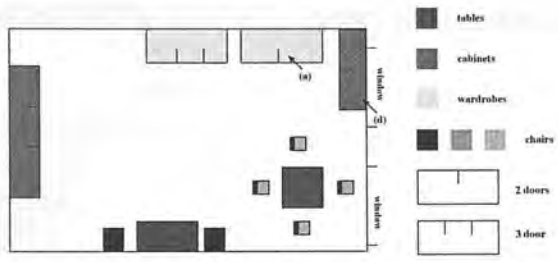


Figure B.18: The screen display for Episode 18 of the Room Arrangement System

The original dialogue

(B.35) **User:** What is the size of this $\nwarrow_{(a)}$ wardrobe? (a)
System: `It is 36cm x 72cm.` (b)
User: What is the size of the cabinet near that icon? (c)
System: `It is 33cm x 86cm.` (d)

The polished dialogue

(B.36) **User:** What is the size of this $\nwarrow_{(a)}$ wardrobe? (a)
System: `36cm x 72cm.` (b)
User: And the cabinet near that icon? (c)
System: `You mean this` $\nwarrow_{(d)}$ `one? It is 33cm x 86cm.` (d)

B.19 Episode 19

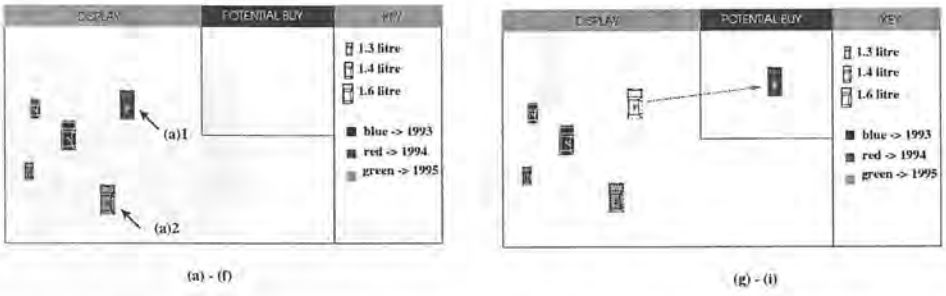


Figure B.19: The screen display for Episode 19 of the Car Selection System

The original dialogue

(B.37)
User: What is the colour of this $\nwarrow_{(a)1}$ car and this $\nwarrow_{(a)2}$ car? (a)
System: `The first car is green, and the second one is white.` (b)
User: What is the price of the green car? (c)
System: `It is 14,200 pounds.` (d)
User: What is the price of the white car? (e)
System: `It is 13,000 pounds.` (f)
Action: *After reading the answer of the system, the user moves the icon of the green car to the potential buy area (see Figure B.19 (g) - (i)).* (g)
User: Give me all information about it. (h)
System: `It is ...` (i)

The polished dialogue

(B.38)
User: What is the colour of this $\nwarrow_{(a)1}$ car and this $\nwarrow_{(a)2}$ car? (a)
System: `The first car is green, and the second one is white.` (b)
User: How much is the green car? (c)
System: `14,200 pounds.` (d)
User: How about the white car? (e)
System: `13,000 pounds.` (f)
Action: *After reading the answer of the system, the user moves the icon of the green car to the potential buy area (see Figure B.19 (g) - (i)).* (g)
User: Give me more information about it. (h)
System: `You already know that the price of this car is 13,000 pounds and it is a white car. The other information is ...` (i)

B.20 Episode 20

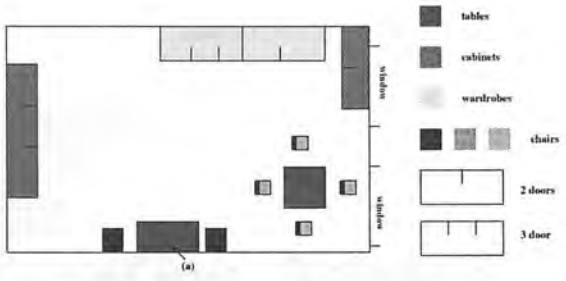


Figure B.20: The screen display for Episode 20 of the Room Arrangement System

The original dialogue

(B.39) **User:** What is the size of this $\nwarrow_{(a)}$ red icon? (a)
System: `It is 33cm x 66cm.` (b)

The polished dialogue

(B.40) **User:** What is the size of this $\nwarrow_{(a)}$ red icon? (a)
System: `Its size is 33cm x 66cm.` (b)

Appendix C

The Evaluation Answer Sheet

The following is the questionnaire we used in the evaluation. Every subject was asked to fill it during the evaluation. The first part is the instruction that indicate the aim of the experiment and the subject's tasks in the experiment. There is no diagram for the graphic displays of the dialogues because the subject is supposed to read the dialogues and the graphic displays from the screen.

An Assessment Questionnaire

Introduction

Thank you for agreeing to help with our study of the *naturalness of human-computer interactions.*

You will be shown fragments of interactions (dialogues) between a user and a computer system, involving both English text and graphical diagrams. We would like you to assess the *naturalness* of the dialogues. You should regard a dialogue as "natural" if it is **fluent and free from misunderstanding between two participants.**

The dialogues

The dialogues are related to two different topics of discussion. One is a *Car Selection System*, and the other is a *Room Arrangement System.*

> The **Car Selection System** is an online version of a catalogue, which is an interactive system displaying details of cars in a stock for sale. These are specific cars, not models of cars; you can think of it as being like a used-car saleroom in that respect. Icons (symbols) in the DISPLAY area represent individual cars; for example, there might be a blue icon representing a particular Nissan 1993 saloon. Various characteristics of the icons convey attributes of the corresponding cars; for example, the colour of the icon might indicate the year of manufacture of that particular car (therefore, the colour of the icons may not be the same as the colour of the cars). A table of how various attributes are represented is displayed in the KEY area. The dialogues involve a user who is browsing through cars with a view to buying. The POTENTIAL BUY area is sometimes used by the user to collect the icons of cars that s/he is interested in. During the interaction, the user can ask about the cars on the screen, or perform actions (such move, remove and add) on the icons of those cars.
>
> The **Room Arrangement System** is an interactive system for helping the user to try out various layouts of a room, for example when planning the re-furnishing of a house. It displays a plan (aerial view) of the room, and icons representing pieces of furniture in the room. The way that different types of icons depict different types of furniture is shown in the keys at the right of the screen display, where *the colour of the icons represent the type of the furniture; the shapes of the icons are the same as that of the furniture; and the short lines inside the icons mark the front side of and the door numbers of the wardrobes.* The user can arrange the layout of the

room by moving the icons of the furniture in the room, so that the screen display will change to show a revised arrangement. During the interaction, the user also can ask for information about the room and the furnishings.

The questionnaire

The questionnaire is divided into *episodes*, each episode containing a dialogue with one or two screen displays.

The bracketed letters at the right hand side of the dialogue (e.g. (a)) allow particular sentences in the dialogue to be indicated. (The same letters should appear both on the questionnaire version of a dialogue, and the text displayed on the screen for that dialogue.) They are used in the following two ways.

1. During the dialogues, the user may point to some part of the screen (using a mouse or some comparable device). These pointing actions are marked in the dialogues for your benefit by arrows accompanied by letters. For example, ↖ (c) in Figure C.0 is our representation of a pointing action which happens in sentence (c) of Dialogue 0, which has (in the screen display) a corresponding sign of ↖ (c) as well. Each arrow sign is placed in the sentence at a position which roughly corresponds to when the pointing action was made.

2. The bracketed letters also indicate the corresponding relations between the sentences and the parts of the diagrams. For example, the diagram part with caption *(a) - (f)* in Figure C.0 is supposed to be what are displayed on the screen when sentences (a) to (f) in Dialogue 0 happen.

Sometimes either the user or the system performs an action on the screen. This action is marked in the dialogue with *Action*, and the sentence following it explains the nature of the action.

After each dialogue, two questions are given. The first one asks about the naturalness of the dialogue. The degree of naturalness of the dialogue should be specified using the following scale:

1 - unnatural

2 - fairly unnatural

3 - so-so

4 - fairly natural

5 - natural

Please read the dialogue and the screen displays carefully. If you think the whole dialogue is free from misunderstanding, circle 3 to 5 according to how natural you think it is. If any part of the dialogue is wrong for the reason of misunderstanding, circle 1 to 2 according to how badly you think the error is.

The second question should be answered if your naturalness rate of the dialogue is either 1 (i.e. unnatural) or 2 (i.e. fairly unnatural). Please circle any part(s) in the dialogue that seem unnatural.

Example

Here is an example. Given the following dialogue, you might complete the questionnaire as follows, because you felt that the System misunderstood the user's phrase "`the green car`" in sentence (e) (i.e. the User intends to indicate the car mentioned in (a) by using that phrase, but the System wrongly picks up the green icon, which represents a different car), the degree of the naturalness is circled to be 1, and, "`the green icon`", the phrase that the System uses in sentence (f), is circled to indicate the reason.

Figure C.0: The screen display for the Example dialogue of the Car Selection System

User:	What is the colour of this $\nwarrow_{(a)}$ car?	(a)
System:	`It is green.`	(b)
User:	Is its insurance group lower than that of this $\nwarrow_{(c)}$ red car?	(c)
System:	`Yes, its insurance group is lower.`	(d)
User:	Move the green car to the potential buy area.	(e)
System:	(`The green icon`) `has been moved.`	(f)

Naturalness: (1) 2 3 4 5

If you choose 1 or 2, please circle on the dialogue the part(s) that seem unnatural:

Now you have completed the instructions. The questionnaire starts from the next page.

C.1 Episode 1

Figure C.1: The screen display for Episode 1 of the Car Selection System

(C.1)
User: Is there a silver car in your stock? (a)
System: `Yes, we have quite a few. Here all the silver cars are displayed on the screen.` (b)
User: Is this $\nwarrow_{(c)}$ car a five-door hatchback? (c)
System: `Yes, it is.` (d)
User: What about the green car near it then? (e)
System: `No, it isn't. It is a three-door hatchback.` (f)
User: Move the green car to the potential buy area. (g)
System: `The green car has been moved.` (h)

- Naturalness: 1 2 3 4 5
- If you choose 1 or 2, please circle on the dialogue the part(s) that seem unnatural.

C.2 Episode 2

Figure C.2: The screen display for Episode 2 of the Room Arrangement System

(C.2)
User: How large is the gap at the left of this $\nwarrow_{(a)}$ cabinet? (a)
System: `You mean this` $\nwarrow_{(b)}$ `gap? It is 18cm.` (b)

- Naturalness: 1 2 3 4 5
- If you choose 1 or 2, please circle on the dialogue the part(s) that seem unnatural.

C.3 Episode 3

Figure C.3: The screen display for Episode 3 of the Room Arrangement System

(C.3)
User: What is the size of this $\nwarrow_{(a)}$ long cabinet? (a)
System: `33cm x 142cm.` (b)
User: How about this $\nwarrow_{(c)}$ short cabinet? (c)
System: `33cm x 86cm.` (d)
Action: *The user moves the long cabinet to the right end of and alongside the short cabinet (see Figure C.3 (e) - (g)).* (e)
User: What is the distance between the cabinet and this $\nwarrow_{(f)}$ table? (f)
System: `The distance between the long cabinet and the table is 93 cm.` (g)

- Naturalness: 1 2 3 4 5
- If you choose 1 or 2, please circle on the dialogue the part(s) that seem unnatural.

C.4 Episode 4

Figure C.4: The screen display for Episode 4 of the Room Arrangement System

(C.4) **User:** How large is the gap at the right of this $\nwarrow_{(a)}$ icon? (a)
System: `You mean this` $\nwarrow_{(b)}$ `gap? It is 2.1cm.` (b)

- Naturalness: 1 2 3 4 5
- If you choose 1 or 2, please circle on the dialogue the part(s) that seem unnatural.

C.5 Episode 5

Figure C.5: The screen display for Episode 5 of the Car Selection System

(C.5) **User:** How much is the green car? (a)
System: `14,200 pounds.` (b)
User: It is too expensive. Remove it from the screen. (c)
System: `Remove can only be used on icons, but your phrase ''it'' refers to a car.` (d)

- Naturalness: 1 2 3 4 5
- If you choose 1 or 2, please circle on the dialogue the part(s) that seem unnatural.

C.6 Episode 6

Figure C.6: The screen display for Episode 6 of the Room Arrangement System

(C.6) **User:** What is the size of this $\nwarrow_{(a)}$ table? (a)
System: `33cm x 66cm.` (b)
User: What is the distance between this $\nwarrow_{(c)}$ cabinet and the wall above it? (c)
System: `You mean this` $\nwarrow_{(d)}$ `wall? The distance is 48cm.` (d)
User: Move the table to the place above the cabinet. (e)
System: `The table has been moved.` (f)

- Naturalness: 1 2 3 4 5
- If you choose 1 or 2, please circle on the dialogue the part(s) that seem unnatural.

C.7 Episode 7

Figure C.7: The screen display for Episode 7 of the Car Selection System

(C.7) **User:** What is the insurance group of this $\nwarrow_{(a)}$ car? (a)
System: `It is in group 5.` (b)
User: What is the insurance group of this $\nwarrow_{(c)}$ red car then? (c)
System: `It is in group 8.` (d)
Action: *The user moves the icon of the car mentioned in (a) to the potential buy area (see Figure C.7 (e) - (g)).* (e)
User: Give me more information about it. (f)
System: `You already know that its insurance group is 5. The other information is ...` (g)

- Naturalness: 1 2 3 4 5
- If you choose 1 or 2, please circle on the dialogue the part(s) that seem unnatural.

C.8 Episode 8

Figure C.8: The screen display for Episode 8 of the Room Arrangement System

(C.8) **User:** What is the distance between this $\nwarrow_{(a)}$ cabinet and the wall to its right? (a)
System: `You mean this` $\nwarrow_{(b)}$ `wall? The distance is 48cm.` (b)

- Naturalness: 1 2 3 4 5
- If you choose 1 or 2, please circle on the dialogue the part(s) that seem unnatural.

C.9 Episode 9

Figure C.9: The screen display for Episode 9 of the Car Selection System

(C.9) **User:** What is the colour of this $\nwarrow_{(a)1}$ car and this $\nwarrow_{(a)2}$ car? (a)
System: `The first car is green, and the second one is white.` (b)
User: Is the green car a five-door hatchback? (c)
System: `Yes, it is.` (d)
User: How about the white car? (e)
System: `No, it is not. It is a saloon.` (f)
User: Move the green icon to the potential buy area. (g)
System: `You mean this` $\nwarrow_{(h)}$ `icon?` *[Action: The icon is moved.]* `It has been moved.` (h)

- Naturalness: 1 2 3 4 5
- If you choose 1 or 2, please circle on the dialogue the part(s) that seem unnatural.

C.10 Episode 10

Figure C.10: The screen display for Episode 10 of the Car Selection System

(C.10) **User:** How much is the green car? (a)
System: `14,200 pounds.` (b)
User: It is too expensive. Remove it from the screen. (c)
System: `The green car has been removed from the screen.` (d)

- Naturalness: 1 2 3 4 5
- If you choose 1 or 2, please circle on the dialogue the part(s) that seem unnatural.

C.11 Episode 11

Figure C.11: The screen display for Episode 11 of the Car Selection System

(C.11) **User:** What is the size of this $\nwarrow_{(a)}$ wardrobe? (a)
System: `36cm x 72cm.` (b)
User: And the cabinet near that icon? (c)
System: `Near which icon? I do not know.` (d)

- Naturalness: 1 2 3 4 5
- If you choose 1 or 2, please circle on the dialogue the part(s) that seem unnatural.

C.12 Episode 12

Figure C.12: The screen display for Episode 12 of the Car Selection System

(C.12) **User:** What is the colour of this $\nwarrow_{(a)1}$ car and this $\nwarrow_{(a)2}$ car? (a)
System: `The first car is green, and the second one is white.` (b)
User: How much is the green car? (c)
System: `14,200 pounds.` (d)
User: How about the white car? (e)
System: `13,000 pounds.` (f)
Action: *After reading the answer of the system, the user moves the icon of the green car to the potential buy area (see Figure C.12 (g) - (i)).* (g)
User: Give me more information about it. (h)
System: `You already know that the price of this car is 14,200 pounds and it is a green car. The other information is ...` (i)

- Naturalness: 1 2 3 4 5
- If you choose 1 or 2, please circle on the dialogue the part(s) that seem unnatural.

C.13 Episode 13

Figure C.13: The screen display for Episode 13 of the Room Arrangement System

(C.13) **User:** What is the distance between this $\nwarrow_{(a)1}$ long cabinet and this $\nwarrow_{(a)2}$ long table? (a)
System: `The distance is 179 cm.` (b)
Action: *The user moves the icon of the short cabinet to the place alongside the long cabinet (see Figure C.13 (c)-(e)).* (c)
User: What is the distance between the cabinet and the long table? (d)
System: `The distance between the long cabinet and the long table is still 179 cm.` (e)

- Naturalness: 1 2 3 4 5
- If you choose 1 or 2, please circle on the dialogue the part(s) that seem unnatural.

Figure C.14: The screen display for Episode 14 of the Car Selection System

C.14 Episode 14

(C.14) **User:** Is there a silver car in your stock? (a)
System: `Yes, we have quite a few. Here, all the silver cars are displayed on the screen for you.` (b)
User: Is this $\nwarrow_{(c)}$ car a five-door hatchback? (c)
System: `Yes, it is.` (d)
User: What about the green car near it then? (e)
System: `In our stock, there is no green car near that five-door hatchback.` (f)

- Naturalness: 1 2 3 4 5
- If you choose 1 or 2, please circle on the dialogue the part(s) that seem unnatural.

C.15 Episode 15

Figure C.15: The screen display for Episode 15 of the Car Selection System

(C.15) **User:** What is the colour of this $\nwarrow_{(a)1}$ car and this $\nwarrow_{(a)2}$ car? (a)
System: `The first car is green, and the second one is white.` (b)
User: Is the green car a five-door hatchback? (c)
System: `Yes, it is.` (d)
User: How about the white car? (e)
System: `No, it is not. It is a saloon.` (f)
User: Move the green icon to the potential buy area. (g)
System: `You mean this` $\nwarrow_{(h)}$ `icon?` *[Action: The icon is moved.]* `It has been moved.` (h)

- Naturalness: 1 2 3 4 5
- If you choose 1 or 2, please circle on the dialogue the part(s) that seem unnatural.

C.16 Episode 16

Figure C.16: The screen display for Episode 16 of the Car Selection System

(C.16) **User:** What is the insurance group of this $\nwarrow_{(a)}$ car? (a)
System: `It is in group 5.` (b)
User: What is the insurance group of this $\nwarrow_{(c)}$ red car then? (c)
System: `It is in group 8.` (d)
User: Give me more information about it. (e)
System: `You already know that its insurance group is 5. The other information is ...` (f)

- Naturalness: 1 2 3 4 5
- If you choose 1 or 2, please circle on the dialogue the part(s) that seem unnatural.

C.17 Episode 17

Figure C.17: The screen display for Episode 17 of the Room Arrangement System

(C.17) **User:** What is the size of this $\nwarrow_{(a)}$ table? (a)
System: `53cm x 79cm.` (b)
User: What is the distance between this $\nwarrow_{(c)}$ cabinet and the wall above it? (c)
System: `You mean this` $\nwarrow_{(d)}$ `wall? The distance is 60cm.` (d)
User: Move the table to the place above the cabinet. (e)
System: `You mean like this` $\nwarrow_{(f)}$`?` (f)

- Naturalness: 1 2 3 4 5
- If you choose 1 or 2, please circle on the dialogue the part(s) that seem unnatural.

C.18 Episode 18

Figure C.18: The screen display for Episode 18 of the Car Selection System

(C.18) **User:** What is the size of this $\nwarrow_{(a)}$ wardrobe? (a)
System: `36cm x 72cm.` (b)
User: And the cabinet near that icon? (c)
System: `You mean this` $\nwarrow_{(d)}$ `one? It is 33cm x 86cm.` (d)

- Naturalness: 1 2 3 4 5
- If you choose 1 or 2, please circle on the dialogue the part(s) that seem unnatural.

C.19 Episode 19

Figure C.19: The screen display for Episode 19 of the Car Selection System

(C.19) **User:** What is the colour of this $\nwarrow_{(a)1}$ car and this $\nwarrow_{(a)2}$ car? (a)
System: `The first car is green, and the second one is white.` (b)
User: How much is the green car? (c)
System: `14,200 pounds.` (d)
User: How about the white car? (e)
System: `13,000 pounds.` (f)
Action: *After reading the answer of the system, the user moves the icon of the green car to the potential buy area (see Figure C.12 (g) - (i)).* (g)
User: Give me more information about it. (h)
System: `You already know that the price of this car is 13,000 pounds and it is a white car. The other information is ...` (i)

- Naturalness: 1 2 3 4 5
- If you choose 1 or 2, please circle on the dialogue the part(s) that seem unnatural.

C.20 Episode 20

Figure C.20: The screen display for Episode 20 of the Room Arrangement System

(C.20) **User:** What is the size of this $\nwarrow_{(a)}$ red icon? (a)
System: `Its size is 33cm x 66cm.` (b)

- Naturalness: 1 2 3 4 5
- If you choose 1 or 2, please circle on the dialogue the part(s) that seem unnatural.

You have finished filling the questionnaire. Thank you very much!